Universidad de Oviedo

Escuela de Ingeniería Informática
Universidad de Oviedo

Proyecto Fin de Máster

Validación de grafos de conocimiento colaborativos utilizando el historial de ediciones

Autor:
Alejandro González Hevia

Director:
Daniel Gayo Avello

Trabajo presentado en cumplimiento de los requisitos para el Máster en Ingeniería Web

Universidad de Oviedo

Escuela de Ingeniería Informática

10 de junio de 2022

Declaración de autoría

El autor del Trabajo Fin de Máster "Validación de Grafos de Conocimiento Colaborativos utilizando el Historial de Ediciones" declara que asume la originalidad del trabajo presentado, habiendo citado debidamente las fuentes utilizadas.

Abstract

Knowledge graphs have been adopted in many diverse fields for a variety of purposes. Most of those applications rely on valid and complete data to deliver their results, pressing the need to improve the quality of knowledge graphs. A number of solutions have been proposed to that end, ranging from rule-based approaches to the use of probabilistic methods, but there is an element that has not been considered yet: the edit history of the graph. In the case of collaborative knowledge graphs (e.g., Wikidata), those edits represent the process in which the community reaches some kind of fuzzy and distributed consensus over the information that best represents each entity, and can hold potentially interesting information to be used by knowledge graph refinement methods. In this work, we explore the use of edit history information from Wikidata to improve the performance of type prediction methods. To do that, we have first built a JSON dataset containing the edit history of every instance from the 100 most important classes in Wikidata. This edit history information is then explored and analyzed, with a focus on its potential applicability in knowledge graph refinement tasks. Finally, we propose and evaluate two new methods to leverage this edit history information in knowledge graph embedding models for type prediction tasks. Our results show an improvement in one of the proposed methods against current approaches, showing the potential of using edit information in knowledge graph refinement tasks and opening new promising research lines within the field.

Resumen

Los grafos de conocimiento han sido adoptados en los últimos años por muchos campos para implementar diversas aplicaciones. La mayoría de estas aplicaciones dependen de que los datos del grafo sean válidos y estén completos para poder ofrecer sus resultados, enfatizando la necesidad de mejorar la calidad de los mismos. Se han propuesto muchos métodos para intentar solucionar este problema, yendo desde enfoques orientados a la definición de reglas hasta otros que utilizan métodos probabilísticos y clasificadores, pero hay un elemento que a día de hoy todavía no ha sido considerado por ninguno de estos métodos: el historial de ediciones del grafo. En el caso concreto de grafos de conocimiento colaborativos, estas ediciones representan el proceso a partir del cuál la comunidad va construyendo un consenso sobre la información que mejor representa a cada entidad, y pueden tener información potencialmente útil para ser aprovechada por métodos de refinamiento de grafos de conocimiento. En este trabajo exploraremos el uso del historial de ediciones de Wikidata para mejorar el rendimiento de las técnicas de predicción de tipos actuales. En primer lugar, construimos un dataset en JSON que contiene el historial de ediciones completo de cada instancia de las 100 clases más importantes de Wikidata. Esta información es posteriormente explorada y analizada, haciendo un énfasis en su potencial aplicabilidad a tareas de refinamiento de grafos de conocimiento. Por último, proponemos y evaluamos dos nuevos métodos de predicción de tipos que hacen uso del historial de ediciones. Los resultados muestran una mejora en uno de los métodos propuestos sobre los enfoques actuales, indicando el potencial de utilizar el historial de ediciones en tareas de refinamiento de grafos de conocimiento y mostrando nuevas áreas de investigación prometedoras.

Agradecimientos

Me gustaría comenzar dándole las gracias a mi familia por el gran esfuerzo que han realizado para que pudiera seguir con mi trayectoria académica y por siempre anteponer mi bienestar al de ellos mismos. Sin ellos nada de esto hubiera sido posible.

A mi tutor, Dani, por el esfuerzo realizado durante todo este proyecto y la guía que me ha ofrecido frente a todos los inconvenientes y dudas que fueron surgiendo a lo largo del trabajo. Tanto en el TFG como en este TFM ha sido un tutor de 10, muy involucrado con el trabajo, y una de las causas de que me intentara superar para presentar el mejor trabajo posible.

A los integrantes del grupo WESO con los que estuve colaborando antes de empezar el Máster y con los que compartí posteriormente clase, en especial a Willy, Pablo, Dani, y Jorge. Pase lo que pase os deseo lo mejor en vuestro futuro, os lo merecéis.

A Carlos, por aquellas noches de conversaciones ayudándome con los aspectos matemáticos que se me atascaban. Ojalá tu TFG salga genial (aunque tenías que haber elegido como tema el teorema del resto chino :).

Por último, muchas gracias a Bea por todo el apoyo durante este último año de Máster y por estar siempre ahí durante los buenos y los malos momentos animándome. Espero que tu TFG salga muy bien y puedas recibir ese descanso que tanto te mereces tú también.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Un grafo de conocimiento es un conjunto de datos representados mediante un modelo de grafo, donde los nodos representan entidades y las aristas las relaciones entre estas entidades [1]. El término ganó relevancia tras la introducción del grafo de conocimiento de Google (Google Knowledge Graph), utilizado internamente para enriquecer algunas funcionalidades del buscador [2]. Otros muchos campos de dominio también han incorporado el uso de grafos de conocimiento en los últimos años para llevar a cabo muchas de sus tareas. Algunos ejemplos concretos serían la gestión de documentos legales [3], la bioinformática [4], el e-learning [5], o la generación automática de pruebas para proyectos software [6], entre muchos otros. Por otro lado, también existen grafos de conocimiento abiertos y de propósito general, siendo algunos de los más relevantes DBpedia [7] y Wikidata [8]. Estas iniciativas han conseguido acercar más la Web Semántica a desarrolladores, consiguiendo que puedan llevar a cabo diversas tareas haciendo uso de estos datos que serían muy difíciles de lograr con otro tipo de medios.

Por lo tanto, asegurar la calidad de estos datos es un objetivo crucial para que todas estas tareas puedan llevarse a cabo de manera exitosa. Varios estudios han surgido que intentan evaluar la calidad de grafos de conocimiento de propósito general, definiendo métricas de calidad que nos permitan evaluar mejor la situación actual de muchos grafos de conocimiento utilizados actualmente [9]. Los resultados señalan la existencia de inconsistencias en las restricciones que debe seguir el grafo y la falta de información, entre otros problemas de calidad [10, 11].

Ante estos problemas, muchas alternativas han sido propuestas a lo largo de los últimos años para mejorar la calidad de grafos de conocimiento. Una primera vertiente se basa en la utilización de sistemas de reglas para definir restricciones que los datos del grafo deben cumplir. Un ejemplo de este enfoque sería el uso de lógica descriptiva (DL) para definir este conjunto de restricciones [12]. También se han definido lenguajes con los que poder representar el esquema que deben seguir determinados nodos de un grafo de conocimiento. Dos ejemplos de estos lenguajes serían Shape Expressions (ShEx) [13] y Shape Constraints Language (SHACL)[1]. En el caso de ShEx, Wikidata cuenta con un apartado especial en el que los autores pueden subir sus esquemas para describir la estructura que deberían seguir las instancias de determinadas clases[2].

Otra vertiente consiste en el uso de métodos probabilísticos y modelos predictivos para refinar los contenidos del grafo. Estos métodos pueden utilizar los propios contenidos del grafo como muestras que el modelo utiliza para detectar inconsistencias dentro del grafo (esto es denominado como *silver standard*[3]), o también utilizar fuentes externas para mejorar la calidad del mismo. Dentro de esta familia de métodos destacan *SDType* y *SDValidate*, dos algoritmos

[1] https://www.w3.org/TR/shacl/

[2] Más información disponible en https://www.wikidata.org/wiki/Wikidata:WikiProject_Schemas.

[3] En comparación con un *gold standard*, en el que se consideran los datos como prácticamente una fuente de

que aprovechan la frecuencia de propiedades del grafo para añadir y validar tipos de las instancias del grafo respectivamente [14]. También es bastante común el uso de redes neuronales y otros modelos predictivos para mejorar la calidad del mismo [15, 16].

Sin embargo, hay un elemento que todavía no ha sido explorado por ninguno de los enfoques mencionados previamente: el historial de ediciones del grafo de conocimiento. Wikidata es un ejemplo concreto de grafo de conocimiento colaborativo, en el que la comunidad va añadiendo y eliminando las tripletas que componen el grafo. Cualquier usuario puede comenzar a editar Wikidata cuando quiera, haciendo que la barrera de entrada sea prácticamente inexistente para nuevos contribuidores. Hasta el instante de redacción de esta memoria se han realizado 1.640 millones de ediciones en Wikidata, a través de las cuales la comunidad va modelando la estructura y contenido semántico de las distintas entidades mediante un proceso estigmérgico, no coordinado y descentralizado del que, no obstante, emerge un consenso colectivo.

1.2. Finalidad del trabajo

La principal finalidad de este trabajo es explorar la posibilidad de **utilizar el historial de ediciones** de un grafo de conocimiento para **refinar la calidad de los datos** del mismo. De cara a este trabajo se utilizará Wikidata como grafo de conocimiento sobre el que desarrollar los experimentos, ya que es uno de los grafos de conocimiento más utilizados actualmente y, además, cuenta con un historial de ediciones disponible sobre el que poder trabajar.

Dado que ésta es un área que no ha sido prácticamente explorada hasta la fecha, una buena parte del trabajo consistirá en entender la información disponible en el historial de ediciones y ver cómo se comporta la comunidad de Wikidata mientras va construyendo el grafo. También se propondrán varios métodos iniciales de refinamiento de grafos utilizando el historial de ediciones, con el fin de ver si esta información tiene potencial para poder mejorar la calidad de los datos del grafo.

Por lo tanto, en este trabajo buscamos sentar las bases dentro del uso del historial de ediciones para el refinamiento de grafos de conocimiento, ver su posible potencial, y detectar nuevas vías de trabajo que serían interesantes de explorar en el futuro. De forma más concreta, las contribuciones del trabajo presentado son las siguientes:

- La creación y publicación de un dataset en JSON que contiene el historial de ediciones completo de cada instancia de las 100 clases más importantes de Wikidata, siguiendo el modelo de datos interno de Wikidata.
- Un análisis de los principales patrones de edición de los contribuidores, ediciones realizadas en cada clase, y conflicto en Wikidata a partir de la información de ediciones obtenida. Esta información se analizará con un énfasis en sus posibles aplicaciones en tareas de refinamiento de grafos de conocimiento.
- La propuesta de dos métodos de predicción de tipos que utilizan el historial de ediciones del grafo: la generación de corrupciones a partir de tripletas eliminadas del grafo, y usar la información del historial de ediciones como etiquetas para un clasificador. Realizamos una evaluación de estos enfoques frente a una serie de líneas base y analizamos el impacto de utilizar el historial de ediciones en ambos enfoques.
- Un dataset RDF conteniendo información del historial de ediciones de Wikidata, siguiendo un modelo de datos personalizado donde cada operación y revisión es serializada en el grafo. Este dataset también se ofrece sin la información de ediciones, y puede servir como una base para medir el impacto de utilizar el historial de ediciones en modelos de refinamiento de grafos de conocimiento.

verdad, los datos *silver standard* pueden contener algunos errores o ser incompletos.

1.3. Estructura de la memoria

El resto de la memoria se estructura de la siguiente forma. En el siguiente capítulo explicamos brevemente algunos de los aspectos teóricos más importantes sobre los que trabaja la memoria, para aquellos lectores que estén interesados en consultarlos (capítulo 2). Tras conocer estos aspectos teóricos, pasamos al repaso del estado de arte actual, cubriendo distintos temas relevantes para nuestro trabajo: el comportamiento de edición dentro de Wikidata, técnicas de refinamiento de grafos utilizadas actualmente, y uso del historial de ediciones de grafos de conocimiento, entre otras cosas (capítulo 3). Una vez entendidos los conceptos teóricos necesarios, ya empezamos a tratar directamente sobre el trabajo presentado, explicando los aspectos de dirección y gestión del mismo (capítulo 4). Posteriormente, en el capítulo 5 se detallan los experimentos realizados durante este trabajo que dan lugar a las contribuciones presentadas. En el capítulo 6 pasamos a analizar e interpretar los resultados obtenidos al realizar los experimentos propuestos. Por último, se describen las conclusiones de este trabajo y se comentan las áreas de trabajo futuro que hemos detectado (capítulo 7).

Esta memoria también cuenta con una serie de apéndices complementarios al texto principal. En estos apéndices se incluye el proceso de selección de revista y el artículo enviado como requisito de este trabajo fin de máster (apéndice A). También se ofrecen apéndices que tratan en detalle algunos aspectos de gestión del proyecto (apéndices B, C y D).

Capítulo 2

Fundamentos Teóricos

2.1. Wikidata

Wikidata[1] es un grafo de conocimiento abierto, colaborativo, y de propósito general que recopila datos estructurados que sirven de soporte a Wikipedia y a cualquier persona que los quiera aprovechar. Estos datos se ofrecen a través de una interfaz de usuario, y también se permite la consulta de los mismos a través del lenguaje SPARQL (**S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage). A continuación explicaremos brevemente los elementos del modelo de datos de Wikidata que se mencionarán a lo largo de la memoria.

2.1.1. Modelo de datos

Las *entidades* son la pieza fundamental sobre la que se sostiene el modelo de datos de Wikidata. Existen dos tipos distintos de entidades: los *objetos* (*items*) y las *propiedades*. Los objetos se corresponden con los elementos que se representan en Wikidata (p.ej., personas, países, ciudades, libros...), y las propiedades se utilizan para añadir información a los objetos (p.ej., fecha de nacimiento de una persona, capital de un país, autor de un libro...). Ambos elementos reciben un identificador numérico único generado de forma auto-incremental. En el caso de los objetos este identificador comienza con la letra 'Q', mientras que para las propiedades comienza con la 'P'. En la figura 2.1 se muestra un extracto de un objeto a través de la interfaz de Wikidata, y los principales elementos que lo componen. A continuación, pasamos a explicar cada uno de estos elementos.

Una *declaración* (*statement*) está compuesta por una propiedad y un valor asignado a esa propiedad. Opcionalmente, puede tener de 1 a n *calificadores* y de 1 a n *referencias*. A lo largo de esta memoria se utilizará el término *declaración simple* para referirnos a declaraciones que tan solo tienen una propiedad y un valor. Un *grupo de declaraciones* es el conjunto de declaraciones de un objeto de una misma propiedad. Cada entidad de Wikidata tiene de 0 a n grupos de declaraciones.

Los *calificadores* se utilizan para dar información adicional sobre una declaración (p.ej., el punto en el tiempo en el que dicha declaración tuvo efecto). Cada calificador también se compone de una propiedad y de un valor asignado a dicha propiedad. La combinación de propiedad, valor, y calificador se denomina en Wikidata como *afirmación*. Las *referencias* también son pares propiedad-valor, y proporcionan una fuente que valida la declaración.

El *fingerprint* de una entidad se compone de las etiquetas, descripciones, y aliases de la misma. La combinación de descripción y etiqueta de una entidad en cada idioma debe de ser única. Una entidad puede tener múltiples aliases, pero tan sólo una descripción y etiqueta para cada idioma.

[1] https://www.wikidata.org/wiki/Wikidata:Main_Page

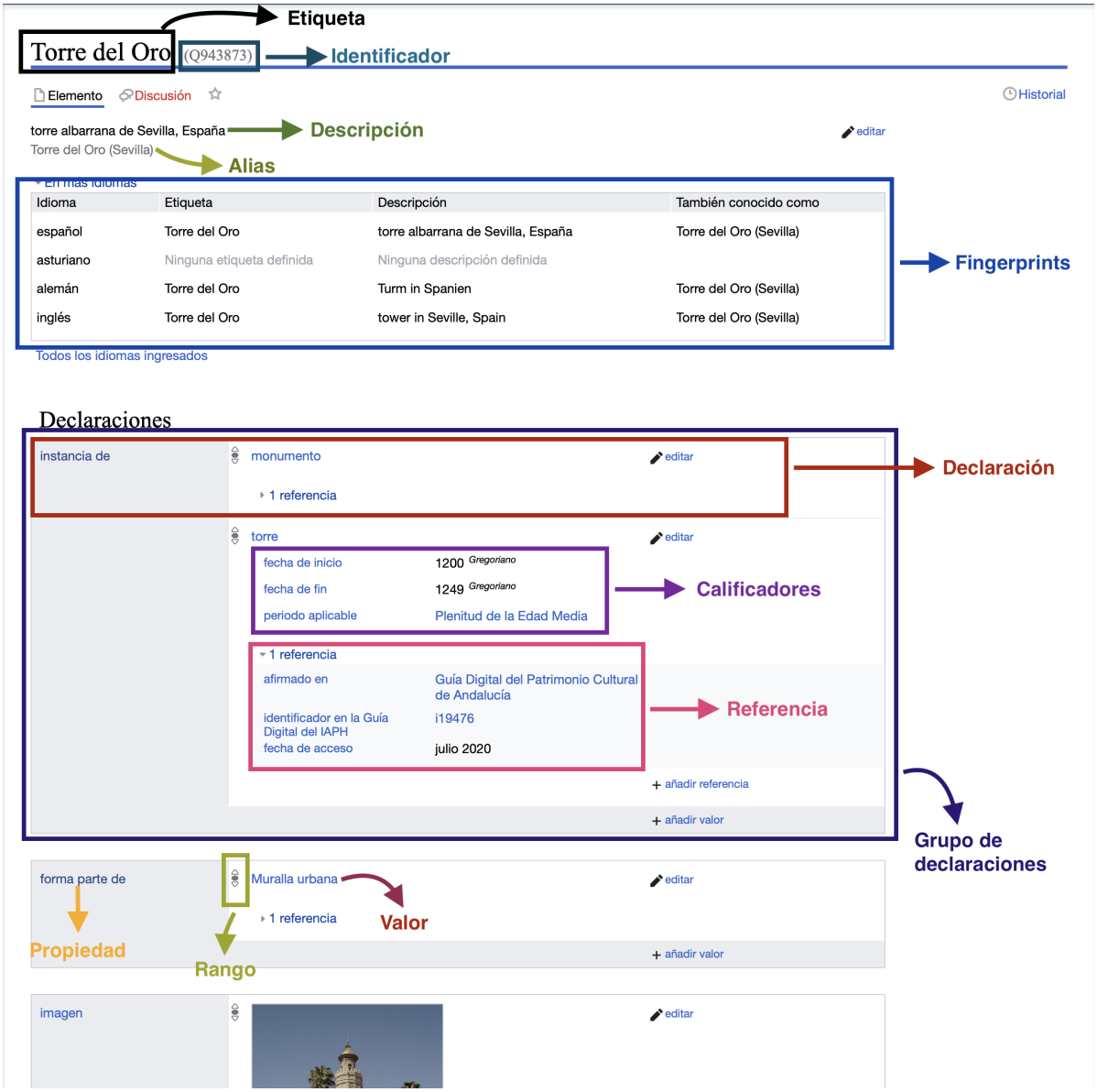


Figura 2.1. Extracto de una entidad de Wikidata y sus elementos.

Los *snaks* son la estructura de información básica de Wikidata, y proporcionan información sobre el valor de una propiedad. Existen tres tipos de snaks: *value*, *somevalue*, y *novalue*. Los snaks de tipo value indican que la propiedad tiene un valor conocido, que es posteriormente representado utilizando uno de los tipos de datos disponibles en Wikidata[2]. Los snaks de tipo somevalue indican que la propiedad tiene un valor desconocido. Por último, los snaks de tipo novalue indican que la propiedad no tiene ningún valor.

Por último, Wikidata también introduce tres *rangos* que pueden ser asignados a cada declaración: `preferido`, `normal`, y `obsoleto`. Estos rangos se suelen utilizar para decidir las declaraciones que se devuelven al realizar consultas sobre Wikidata, y también para limpiar y organizar la interfaz de usuario cuando se explora una entidad. Las declaraciones también pueden tener un orden dentro de cada grupo de declaraciones. Aunque el orden de las declaraciones dentro de cada rango no es relevante, éste puede ser modificado por los usuarios.

Todos estos elementos son serializados internamente a JSON y RDF. Aunque hemos cu-

[2]De cara a este trabajo no es necesario conocer estos tipos de datos. Para más información, se puede consultar el siguiente enlace: https://www.wikidata.org/wiki/Special:ListDatatypes

bierto toda la lista de elementos que serán mencionados durante la memoria, esta lista no es exhaustiva[3].

2.1.2. Historial de revisiones

Wikidata permite la edición de entidades por cualquier usuario. Cada entidad está compuesta de un *historial de revisiones*, que contiene todos los cambios realizados a la entidad por los editores. Cualquiera de los elementos que hemos mencionado en la sección anterior puede ser modificado, y una única revisión puede contener cualquier número de cambios sobre cualquier combinación de elementos de una entidad. A lo largo de esta memoria utilizaremos los términos *edición* y *revisión* de forma intercambiable.

Una revisión se compone de una serie de metadatos adicionales, entre los que se incluye la fecha de realización, el autor, etiquetas, y una descripción. Las etiquetas se suelen utilizar para indicar el dispositivo desde el que se ha realizado la revisión (o la herramienta utilizada, en caso de ser una edición automatizada) y para indicar la causa de la edición[4] (p.ej., para revertir un acto de vandalismo).

En el contexto de este trabajo usaremos el término *operación* para referirnos a una modificación realizada sobre un elemento de una entidad en una revisión. Una revisión está compuesta de 1 a n operaciones, donde cada operación puede representar el añadido o eliminado de un elemento de la entidad afectada. También tendremos en cuenta a lo largo de la memoria las operaciones de modificación de elementos, que pueden ser consideradas como la combinación de una operación de eliminado y otra de añadido sobre el elemento.

Wikidata permite *deshacer* y *restaurar* revisiones realizadas sobre una entidad. La restauración permite revertir todas las revisiones que se realizaron sobre una entidad hasta el instante de tiempo seleccionado. El proceso de deshacer es más versátil, ya que permite revertir de 1 a n revisiones seleccionadas por el usuario, que no tienen por que ser revisiones sucesivas como en el caso de la restauración. En ninguno de estos casos se eliminan las revisiones que se están revirtiendo del historial de ediciones del grafo. Lo que se hace es crear una nueva revisión, que contiene todas las operaciones necesarias para revertir las revisiones seleccionadas por el usuario.

Cada revisión puede ser eliminada manualmente por administradores de Wikidata bajo ciertas circunstancias. Entre estos casos especiales se encuentran eliminar revisiones que contengan información privada, una violación de copyright, o ataques personales.

2.2. Modelos basados en embeddings

Una ventaja de los datos estructurados que ofrece Wikidata es que estos pueden ser modelados en forma de grafo. En la figura 2.2 mostramos un grafo construido a partir de las declaraciones simples de la entidad que hemos explorado en la sección anterior. Este grafo puede ser serializado posteriormente a distintos formatos que lo representen. Como hemos comentado anteriormente, en el caso de Wikidata estos datos son serializados internamente a RDF[5], un modelo de datos basado en tripletas. Cada tripleta $t = (s, p, o)$ está compuesta por un sujeto o nodo origen (s), un objeto o nodo destino (o), y la relación que une el sujeto y objeto, llamada predicado (p).

[3] Para más información sobre el modelo de datos de Wikidata se recomienda consultar el siguiente enlace: https://www.mediawiki.org/wiki/Wikibase/DataModel

[4] A través del siguiente enlace se puede ver una lista de las etiquetas más comunes utilizadas: https://www.wikidata.org/wiki/Special:Tags

[5] https://www.w3.org/RDF/

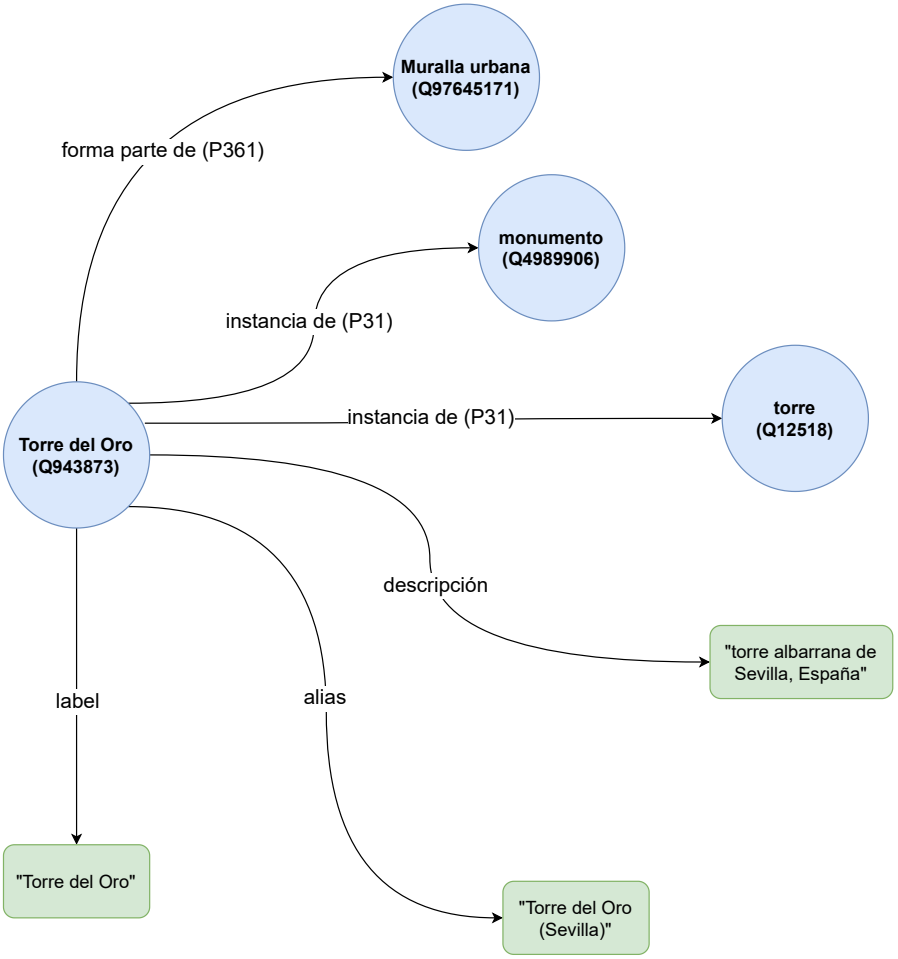


Figura 2.2. Grafo construido a partir de los elementos básicos de la entidad vista previamente. Los círculos representan objetos existentes en Wikidata y los rectángulos tipos de datos simples.

Nota: Representando elementos complejos en el grafo de Wikidata

Wikidata serializa todos los elementos de su modelo de datos a RDF, incluidos los calificadores y referencias. Dado que estos últimos elementos son tripletas que hablan sobre otras tripletas (p.ej., el instante de tiempo en el que una tripleta ocurrió), Wikidata utiliza una técnica denominada *reificación*[a] para poder representarlas en el grafo.
De cara a este trabajo se utilizarán tan sólo declaraciones simples sin reificar, ya que los métodos de *embeddings* actuales todavía están evolucionando para intentar tratar con ese tipo de serialización.

[a]Para más información sobre esa técnica se recomienda la lectura del siguiente documento: https://aidanhogan.com/docs/reification-wikidata-rdf-sparql.pdf

El hecho de trabajar con grafos nos permite utilizar técnicas de análisis de datos específicas de estos modelos de datos. Una de ellas es el uso de *knowledge graph embeddings* (*KGE*). El objetivo de esta técnica es obtener una representación vectorial de cada uno de los nodos y vértices del grafo. Existen distintos modelos de *knowledge graph embeddings*, donde cada modelo define una función $f_p(s, o)$ que mide la probabilidad de que una tripleta $t = (s, p, o)$ sea verdadera, y optimiza la función a partir de los datos del grafo para calcular los vectores de cada elemento.

Aunque definir los distintos tipos de modelos existentes queda fuera del alcance de esta

memoria, existen principalmente dos variantes. La primera son los modelos que utilizan redes neuronales para aprender estos vectores, en un proceso similar a modelos más conocidos en el área del procesamiento del lenguaje natural como *word2vec*[6]. En esta primera variante destacan los modelos *RDF2Vec* [17] y *ConvKB* [18]. La segunda variante intenta minimizar la distancia entre los vectores del sujeto y del objeto al ser proyectados por una matriz asignada a cada predicado. Dentro de esta segunda variante destacan los modelos *TransE* [19] y *TransR* [20].

La interpretación a nivel semántico de los vectores obtenidos puede variar en función del tipo de modelo y de función f_p definida. Una vez que obtenemos estos vectores, se pueden usar de entrada a otras técnicas de aprendizaje automático como el clústering o la clasificación. Por lo tanto, son bastante utilizados en muchos sistemas de refinamiento de grafos de conocimiento.

2.2.1. Muestreo negativo

Todos los modelos de *knowledge graph embeddings* mencionados previamente necesitan tener una serie de tripletas positivas y negativas para optimizar su función f_p particular. En el caso de las tripletas positivas se asume que todas las tripletas pertenecientes al grafo de conocimiento con el que estamos trabajando son positivas.

Para obtener el conjunto de tripletas negativas se lleva a cabo un proceso conocido como *muestreo negativo*, en el que se generan una serie de corrupciones (tripletas negativas) a partir de cada tripleta positiva del grafo. Actualmente una de las técnicas más usadas para generar corrupciones es modificar el sujeto o el objeto de cada tripleta positiva con un nodo aleatorio del grafo de conocimiento, bajo la premisa de que el espacio de combinaciones de sujeto, predicado, objeto posibles es mucho mayor que el de tripletas verdaderas existentes. Por lo tanto, se asume que la probabilidad de insertar falsos negativos en este proceso es muy baja.

Estas muestras negativas generadas se juntan con las tripletas del grafo para optimizar la función f_p del modelo que estemos utilizando y obtener la representación vectorial de cada elemento del grafo.

Conclusiones

En este capítulo hemos definido los elementos principales a nivel teórico que se irán utilizando a lo largo de la memoria. Se trabajará sobre Wikidata, un grafo de conocimiento colaborativo de propósito general, y se utilizarán métodos basados en *embeddings* que calculan representaciones vectoriales de cada uno de los elementos del grafo. Con esta nueva información podemos ahora pasar a la revisión de literatura actual para conocer cuáles han sido los avances en los campos que afectan al trabajo.

[6] De hecho, el proceso en muchos casos es bastante similar, calculando la probabilidad de que aparezca cada objeto a partir de un sujeto y predicado dados.

Capítulo 3

Revisión de Literatura

La revisión de literatura que presentamos se divide en tres áreas principales. Empezaremos describiendo los estudios que buscan entender como se desarrollan las dinámicas de edición en grafos de conocimiento colaborativos. A continuación, revisaremos el trabajo existente en el área de la validación de grafos de conocimiento, comparando los diversos enfoques y métodos existentes. Por último, veremos el uso que se hace del historial de ediciones de los grafos de conocimiento en distintas áreas de la web semántica.

3.1. Entendiendo las dinámicas de edición en Wikipedia y Wikidata

Uno de los primeros estudios sobre la colaboración en Wikidata fue realizado por T. Steiner. El autor implementa y ofrece una herramienta que proporciona datos en tiempo real sobre las estadísticas de edición en Wikipedia y Wikidata [21]. Posteriormente, realiza un análisis cuantitativo sobre las ediciones en Wikipedia y Wikidata a lo largo de 3 días en 2013. Durante ese período de tiempo Wikidata ya era un sistema donde la mayor parte de las ediciones provenían de bots (alrededor de un 88 %).

Mientras que el análisis cuantitativo de Steiner permitía entender la distribución de editores en Wikidata, seguía siendo necesario un análisis cualitativo para saber **qué editaba cada usuario** y poder entender cómo se comportaba cada pieza dentro de este ecosistema. Este estudio lo realizaron Müller-Birn et al. en 2015 [22], recogiendo datos de ediciones de Wikidata desde octubre de 2012 —fecha origen de la base de conocimiento— hasta octubre de 2014, categorizando los datos según el tipo de edición realizada, y agregando estos datos en ventanas de 1 mes para cada colaborador. Con esta información consiguieron detectar seis patrones de edición en Wikidata: editor de referencias, creador de entidades, experto de entidades, editor de entidades, editor de propiedades, e ingeniero de propiedades. Se puede ver cómo Wikidata combina características de un sistema de colaboración en masa, en el que los usuarios van adquiriendo roles dentro del ecosistema de colaboración a partir de la experiencia, con un enfoque de ingeniería de ontologías, en el que un grupo reducido de gente autorizada crea nuevas propiedades y discute sus ámbitos de aplicación. Aún así, el estudio se realizó en una etapa temprana de Wikidata[1], y en el propio artículo se explica que estos patrones de colaboración pueden ir evolucionando a la vez que el propio ecosistema de Wikidata va madurando.

Otro aspecto de interés dentro de este campo es comprender el camino que un nuevo editor de Wikidata sigue hasta convertirse en un contribuidor experimentado. Piscopo et al. detectaron que Wikidata tiene una barrera de entrada muy pequeña, lo que permite a nuevos usuarios empezar a hacer contribuciones sobre declaraciones simples nada más empezar [23].

[1] Por ejemplo, no se podían añadir *calificadores* a una propiedad en aquellos tiempos.

Sin embargo, a medida que los editores se vuelven más expertos alrededor del ecosistema de Wikidata, transicionan de realizar ediciones a través de la interfaz de usuario a utilizar herramientas automatizadas más avanzadas. Entender desde una etapa temprana qué editores acabarán estando más involucrados en realizar ediciones sobre Wikidata fue el objetivo de Sarasua et al. [24]. Los autores monitorizaron la forma en la que los usuarios de Wikidata participaban en la plataforma durante 4 años, y desarrollaron modelos predictivos que estiman la duración durante la cuál un usuario editaría Wikidata, alcanzando una medida F1 de 0,9.

Los trabajos más recientes se han centrado en entender cómo estas dinámicas de colaboración afectan a la calidad del grafo de conocimiento. Piscopo et al. analizaron varias métricas de edición de 5.000 entidades en Wikidata para entender estas relaciones [25]. Los autores detectaron que la interacción entre editores humanos y bots es necesaria para asegurar la calidad de estas entidades, mientras que las ediciones anónimas son perjudiciales para la calidad (principalmente debido a vandalismo). En su trabajo posterior, los autores intentan buscar una correlación entre un conjunto de métricas de calidad definidas y roles de edición detectados con métodos de clústering [26]. Los resultados muestran que Wikidata está muy desigualmente distribuida a nivel ontológico, con algunos dominios teniendo una gran cantidad de subclases creadas por bots.

3.2. Predicción de tipos en grafos de conocimiento

En cuanto a los trabajos relacionados con la tarea de predecir tipos en un grafo de conocimiento, SDType fue una de las primeras herramientas desarrolladas para añadir declaraciones con tipos incompletos en grafos de conocimiento [14] –específicamente, en DBPedia. SDType calcula la distribución estadística de tipos que aparecen tanto de sujeto como de objeto de una propiedad y utiliza esta información para rellenar los tipos de una entidad, mejorando el rendimiento de otros enfoques de predicción de tipos en aquel momento. Melo et al. propusieron posteriormente el uso de un clasificador jerárquico para aprovechar la naturaleza jerárquica de los sistemas de tipos existentes en grafos de conocimiento [27]. Este sistema fue probado en varios grafos de conocimiento, incluido Wikidata, mejorando el rendimiento de SDType y de otras alternativas.

El uso de *embeddings* de grafos de conocimiento también ha sido explorado recientemente. Kejriwal et al. propusieron un método que permitía crear *embeddings* de clases del grafo a partir de una serie de vectores de entidades precalculados [28]. Los vectores obtenidos para cada clase se encuentran más cercanos de las instancias y clases relacionadas en el espacio vectorial resultante. Los autores propusieron posteriormente el uso de la similitud coseno entre el vector de una entidad y los de cada clase para recomendar el tipo de la entidad. Otros enfoques intentan generar predicciones de tipo de grano fino utilizando *embeddings* de entidades tanto con modelos supervisados como no supervisados, obteniendo mejores resultados que SDType en dos datasets de DBPedia [29].

Ridle es una sistema que utiliza redes neuronales y una representación a medida del grafo de conocimiento para mejorar los enfoques de predicción de tipos existentes [30]. La representación de cada entidad propuesta se construye a partir de sus relaciones, bajo la idea de que algunos conjuntos de propiedades son más probables de aparecer como predicados en las instancias de una clase que en otras. Ridle mejora el rendimiento de SDType y de otros métodos basados en *embeddings* en varios datasets, incluyendo Wikidata.

3.3. Muestreo de tripletas negativas

El muestreo de tripletas negativas ha sido demostrado como un proceso importante en el rendimiento final de los modelos basados en *embeddings* de grafos de conocimiento. Kotnis et al. propusieron dos nuevos métodos de muestreo de tripletas negativas basados en *embeddings*,

comparando su rendimiento con enfoques aleatorios [31]. Sus resultados muestran que el rendimiento de cada método de muestreo de tripletas negativas depende de las características del dataset sobre el que se evalúan. Ninguno de los enfoques evaluados fue el mejor en todos los datasets.

El uso de redes generativas antagónicas (GANs en inglés) para generar tripletas negativas también ha sido introducido recientemente, mostrando resultados prometedores al mejorar el rendimiento de los modelos de *embeddings* de grafos de conocimiento bajo diferentes circunstancias [32, 33]. NSCaching es un método que mejora la eficiencia de los métodos basados en GANs al incorporar una caché que contiene las tripletas negativas de mayor calidad, mejorando el rendimiento de otros métodos de muestreo negativo actuales [34, 35].

3.4. Utilización del historial de ediciones

El uso del historial de ediciones de un grafo de conocimiento para diversas tareas ha ido ganando bastante interés recientemente. Los primeros trabajos existentes buscaron formas de facilitar el acceso y la consulta de datos de ediciones de Wikidata. El servicio de consultas del historial de ediciones de Wikidata (Wikidata Edit History Query Service) fue una de las primeras herramientas disponibles con tal fin, proporcionando un *endpoint* de SPARQL sobre el que se podían consultar las ediciones realizadas en Wikidata hasta el 1 de julio de 2018 [36].

Wikidated es un dataset que contiene el historial de ediciones completo de Wikidata, almacenado como un conjunto de añadidos y eliminados de tripletas [37]. Proporciona una API en Python para explorar e interactuar con el dataset sin necesidad de saber el modelo de datos interno utilizado. A fecha de escritura de esta memoria todavía se encuentra en fase de desarrollo.

En cuanto a la aplicación del historial de ediciones de un grafo de conocimiento para tareas de refinamiento de grafos de conocimiento, AlGhamdi et al. propusieron un sistema de recomendación de Wikidata que hace uso de esta información [38]. El sistema calcula estadísticas de las actividades de edición del usuario y combina esto con representaciones de cada entidad para recomendar aquellas entidades que el usuario podría editar a continuación. Los resultados del sistema son prometedores y sientan las bases para trabajo futuro en sistemas de recomendación para Wikidata.

Tanon et al. propusieron un sistema que automáticamente mina reglas de corrección de Wikidata a partir de su historial de ediciones [39]. El sistema parte de un conjunto de violaciones de restricciones que ocurrieron en el grafo, y mina el historial de ediciones para encontrar ediciones que hayan corregido cada violación. Ejecutan una serie de experimentos en Wikidata, mostrando mejoras significativas sobre las líneas base. En su posterior trabajo, los autores proponen el uso de una red neuronal para predecir las reglas de corrección y mejorar los resultados del sistema anterior [40]. Esta red neuronal recibe como entrada una representación vectorial de la restricción, las tripletas que han sido violadas, y hechos sobre cada entidad sacados de fuentes externas. Los resultados sobre Wikidata muestran mejoras significativas sobre el enfoque previo orientado a reglas.

Conclusiones

En este capítulo hemos repasado la literatura existente que más directamente afecta a nuestro proyecto. Hemos visto el interés reciente en entender cómo cada miembro de la comunidad se comporta y coordina con otros dentro de Wikidata, así como las distintas herramientas de validación de grafos que han ido surgiendo durante los últimos años. Con todo esto cubierto, procederemos a explicar cómo se ha gestionado el proyecto a continuación.

Capítulo 4

Dirección y Gestión del Proyecto

Antes de exponer el trabajo realizado y los resultados obtenidos, vamos a presentar las labores de gestión del proyecto que se han llevado a cabo. Todas las actividades de gestión realizadas se basan en la Guía de los fundamentos para la dirección de proyectos (PMBOK). Cada actividad ha sido seleccionada —y en algunos casos también adaptada— buscando como fin la ejecución del proyecto en **tiempo**, cumpliendo con los **objetivos** definidos y con la mayor **calidad** posible.

4.1. Planificación del proyecto

4.1.1. Identificación de interesados

En la tabla 4.1.1 se muestra el registro de interesados de este proyecto. Dentro de los interesados detectados hay cuatro grupos de los cuales no se sabía su identidad mientras se planificaba y desarrollaba el proyecto: los posibles lectores del artículo creado, los usuarios del sistema de validación que se iba a desarrollar, los revisores de la revista donde se enviaría el artículo, y los miembros del tribunal del TFM. En el caso de los tres primeros grupos esto no afectó mucho al alcance o planificación, ya que independientemente del usuario final los intereses generales de cada grupo son relativamente similares. Sin embargo, en el caso de los miembros del tribunal del TFM no fue posible concretar el área de interés ya que puede variar en función de su área de conocimiento[1]. Por lo tanto, se les estableció un área de interés genérico en esta identificación de interesados y a la hora de planificar el proyecto se intentaron abarcar todos estos intereses de la mejor manera posible.

Cuadro 4.1. Tareas relativas al inicio del proyecto

Nombre	Cargo	Función	Área de interés
Alejandro González Hevia	Alumno	Encargado de llevar a cabo el proyecto.	Interesado en que el proyecto cumpla los requisitos exigidos en el Trabajo Fin de Máster en los plazos definidos.

[1] Alguien puede tener más interés en la propia gestión del proyecto, en la metodología investigadora, en aspectos técnicos de la web semántica...

Cuadro 4.1. Tareas relativas al inicio del proyecto

Nombre	Cargo	Función	Área de interés
Daniel Gayo Avello	Director del proyecto	Encargado de guiar al alumno ante los posibles problemas o dudas que surjan en el proyecto.	Interesado en el buen avance del proyecto y desempeño del alumno.
–	Usuarios del sistema de validación	Personas que vayan a ejecutar el sistema de validación desarrollado para sus propios casos de uso.	Interesados en que el sistema se encuentre debidamente documentado y que tenga un buen rendimiento comparado con otras alternativas (tanto en tiempo como en resultados).
–	Lectores del artículo	Personas que vayan a leer el artículo con diversos fines (p.ej., para hacer una revisión de literatura).	Interesados en que el artículo sea fácil de entender, completo, y correcto metodológicamente hablando. También están interesados en la reproducibilidad de los experimentos mostrados.
–	Revisores de la revista asignados	Encargados de evaluar el artículo enviado a la revista JCR para su posible publicación.	Interesados en que la investigación presentada sea innovadora y significante en el área de conocimiento, para tener el mayor impacto posible en caso de ser publicada en su revista.
–	Miembros del tribunal evaluador	Encargados de evaluar el proyecto presentado por el alumno.	Interesados en la calidad del proyecto presentado y el desempeño del alumno.

4.1.2. OBS y PBS

OBS

La estructura organizativa de cara al proyecto presentado se muestra en la figura 4.1. Las responsabilidades del alumno incluyen el desarrollo completo del proyecto (artículo, memoria, experimentos, ...), mientras que el director de proyecto se encarga de ayudar al alumno ante dudas y sugerir enfoques metodológicos de la investigación que poder aplicar. Las tareas rea-

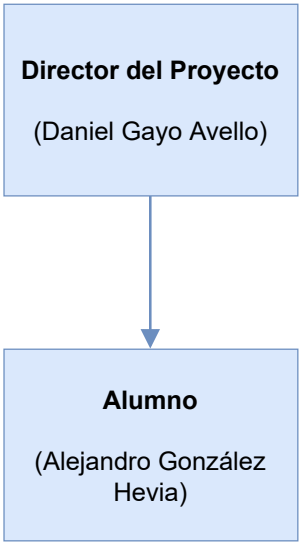


Figura 4.1. Estructura organizacional del proyecto

lizadas por el alumno tienen una supervisión periódica del director y los entregables finales (artículo y memoria) requieren de su aprobación.

PBS

En la figura 4.2 se muestran los principales productos generados en este proyecto. Cada una de las tareas definidas en la planificación del proyecto tiene como resultado un producto. De cara al PBS mostrado en esta sección se seleccionaron los productos principales, pero la lista completa de productos resultantes de cada tarea se puede consultar en el archivo '*planificacion_final.mpp*' adjunto a la entrega.

4.1.3. Planificación inicial (WBS)

En los siguientes apartados vamos a mostrar las labores de gestión del tiempo realizadas.

Definición de tareas

Vamos a pasar ahora a mostrar de forma resumida las tareas que hemos definido para el proyecto. Para cada tarea se indica su identificador, nombre, la duración estimada y el identificador de sus tareas predecesoras —si las tiene—. Mostraremos sólo los tres primeros niveles de jerarquía (tareas, subtareas y 'subsubtareas') a lo largo del documento. Para una visualización completa de las tareas definidas se proporciona la planificación inicial como material complementario a la memoria.

Un pequeño detalle de las tareas mostradas es que la estimación en horas incluye el tiempo necesario para escribir en la memoria el contenido correspondiente a cada tarea. Hemos seguido un enfoque en el que una tarea no podría darse por finalizada hasta haber escrito lo relativo a la tarea en la memoria —o en otros documentos complementarios, cuando proceda—. Las dos opciones posibles eran doblar el número de tareas, añadiendo una tarea adicional de documentación por cada tarea definida, o tener en cuenta un tiempo extra en cada tarea para contemplar la documentación de ésta. En nuestro caso hemos optado por la última opción, teniendo en cuenta que todas las tareas definidas conllevan una documentación de las mismas en esta memoria.

Las tareas definidas para el inicio del proyecto se muestran en la Tabla 4.2. Estas tareas están principalmente relacionadas con la propia gestión del proyecto: estimación de tiempos,

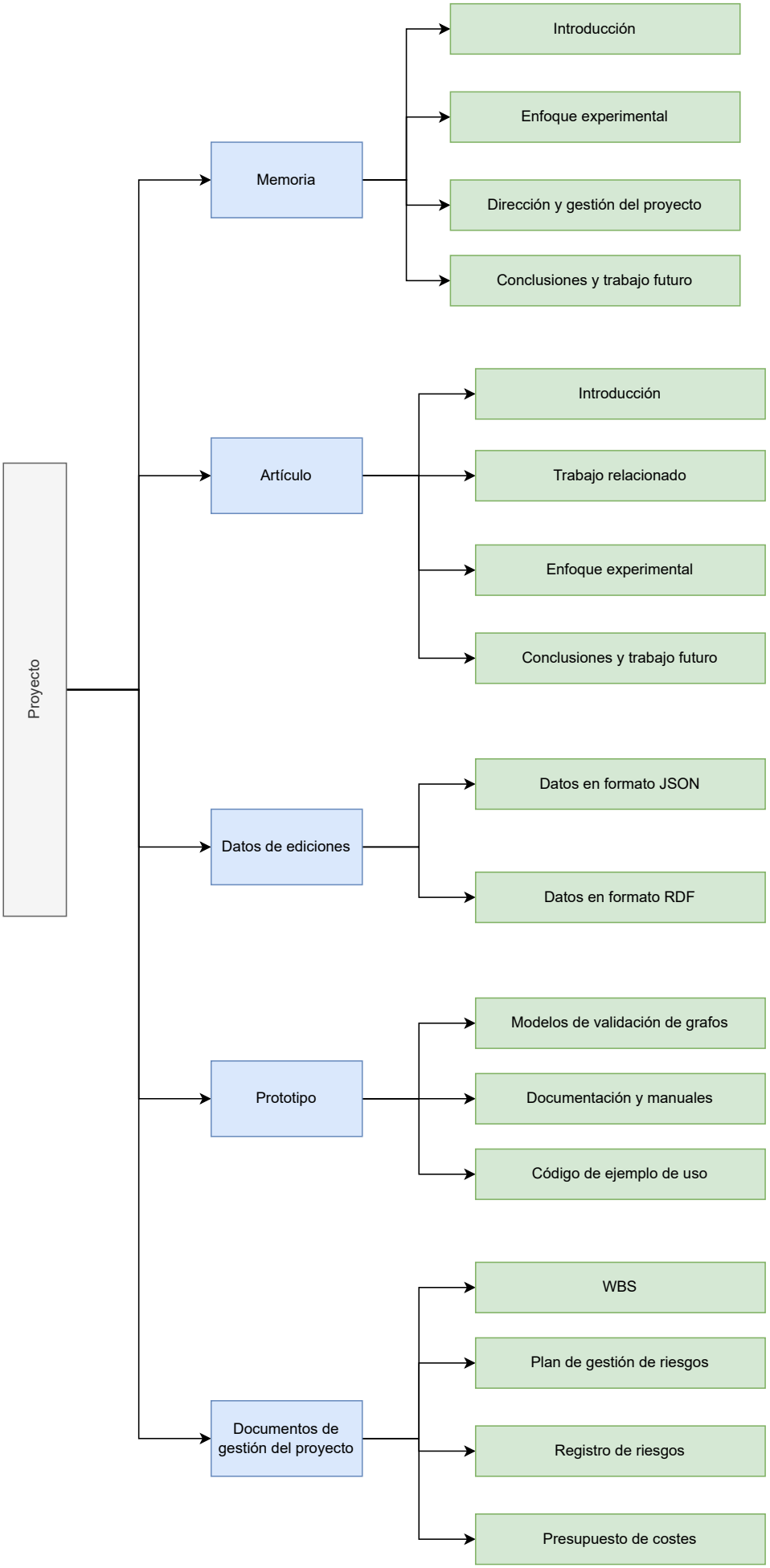


Figura 4.2. Productos principales generados en el proyecto

gestión de riesgos, creación del presupuesto, fijación de objetivos y alcance del proyecto... Esta primera fase estuvo programada desde el **1 de septiembre** hasta el **18 de octubre**, con una duración estimada de **89.4 horas**.

Cuadro 4.2. Tareas relativas al inicio del proyecto

ID	Nombre	Estimación (h)	Fecha fin (dd-mm-yy)	Predecesor
1	**Inicio del proyecto**	**89,4**	**18-10-21**	-
1.1	Definición de objetivos y alcance del proyecto	8,8	02-09-21	-
1.2	Creación de acta de proyecto	8,8	22-09-21	-
1.3	Desglose en tareas	12,4	03-09-21	1.1
1.4	Estimar duraciones de las tareas	10,4	07-09-21	1.3
1.5	Estructura inicial de la memoria	8,2	24-09-21	-
1.6	**Gestión de riesgos**	**18,8**	**30-09-21**	1.1
1.6.1	Definición de lista de riesgos	12,2	29-09-21	-
1.6.2	Estudio de medidas a adoptar	4,4	30-09-21	1.6.1
1.6.3	Refinar tareas con las medidas adoptadas	2,2	30-09-21	1.6.2
1.7	**Creación de presupuesto inicial**	**22**	**18-10-21**	1.4
1.7.1	Cálculo de costes del proyecto	4,8	15-10-21	-
1.7.2	Creación de presupuesto interno	12,8	14-10-21	-
1.7.3	Creación de presupuesto externo (cliente)	4,4	18-10-21	-

La siguiente tanda de tareas se corresponde con el propio desarrollo de la investigación. Dado que es un bloque bastante grande de tareas (más de 700h estimadas) lo vamos a dividir en 3 partes que analizaremos por separado.

La primera parte de la investigación se corresponde con el repaso de la literatura y la obtención de los datos que se utilizan en la propia investigación. Las tareas de esta parte se muestran en la Tabla 4.3 y el diagrama de Gantt correspondiente en la Figura 4.4. Estas tareas se estima que se desarrollen desde el **7 de septiembre** hasta el **11 de enero**.

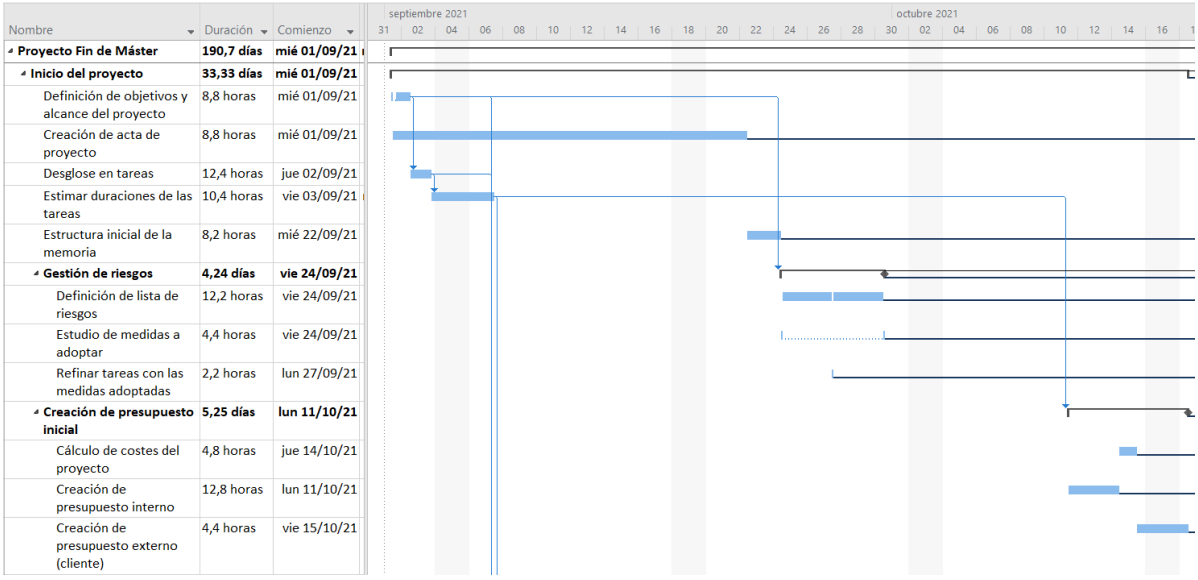


Figura 4.3. Diagrama de Gantt de las tareas relativas al inicio del proyecto

Cuadro 4.3. Tareas relativas al desarrollo de la investigación (I)

ID	Nombre	Estimación (h)	Fecha fin (dd-mm-yy)	Predecesor
2	**Desarrollo de la investigación**	**742,7**	18-05-22	1.1, 1.3, 1.4
2.1	**Revisión de literatura y estado del arte**	**129,2**	05-11-21	-
2.1.1	Uso del historial de revisiones en grafos de conocimiento	24,4	08-10-21	-
2.1.2	Algoritmos de corrección de grafos de conocimiento	48,0	20-09-21	-
2.1.3	Dinámicas de conflicto y colaboración en Wikipedia y Wikidata	32,4	28-10-21	-
2.1.4	Métodos de análisis de grafos	24,4	05-11-21	
2.2	**Obtención y tratamiento de datos**	**152,0**	11-01-22	2.1
2.2.1	Selección de subconjunto de datos a utilizar	25,2	15-11-21	-
2.2.2	Obtención de subconjunto de datos	42,8	30-11-21	2.2.1
2.2.3	Filtrado de datos (p.ej., vandalismo)	28,0	09-12-21	2.2.2
2.2.4	Preprocesamiento	12,8	14-12-21	2.2.3
2.2.5	Transformación de datos no estructurados a tripletas	26,4	21-12-21	2.2.4

Cuadro 4.3. Tareas relativas al desarrollo de la investigación (I)

ID	Nombre	Estimación (h)	Fecha fin (dd-mm-yy)	Predecesor
2.2.6	Creación de grafos de ediciones con los datos obtenidos	16,8	11-01-22	2.2.5

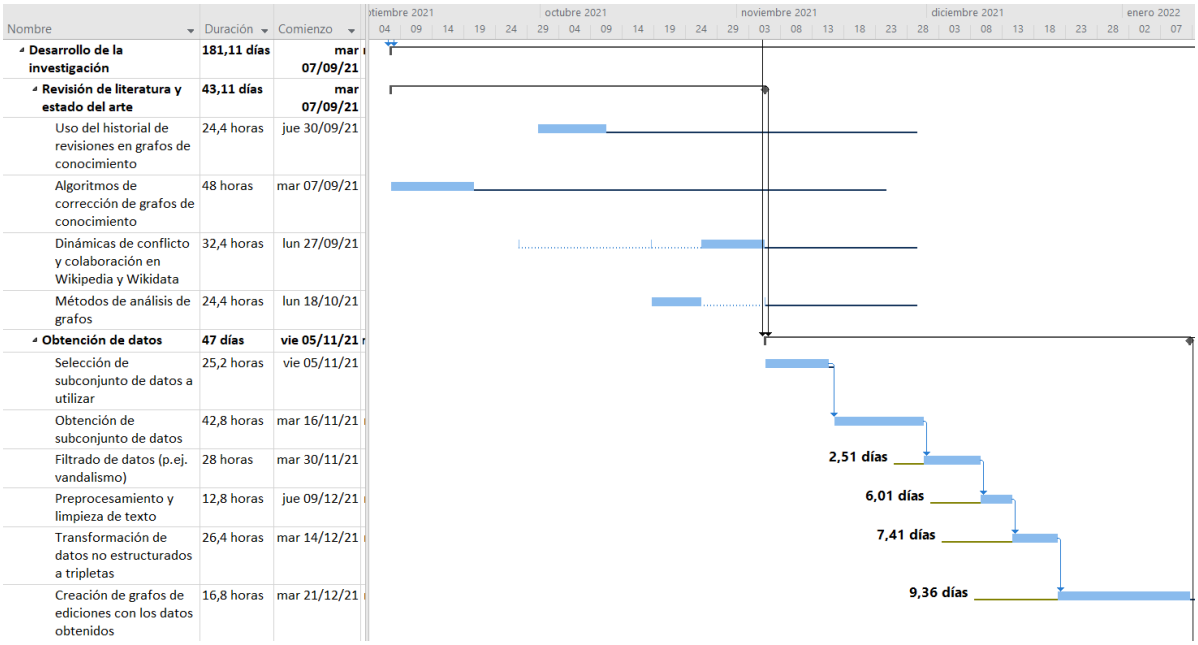


Figura 4.4. Diagrama de Gantt de las tareas relativas al desarrollo de la investigación (I)

A continuación se muestran en la Tabla 4.4 las tareas que forman el núcleo de la investigación. En ellas se incluye la búsqueda de patrones de edición en los datos obtenidos, y su uso para el refinamiento de grafos de conocimiento. Estas tareas duran **270.4 horas**, desarrollándose desde el **10 de enero** hasta el **8 de abril**.

Cuadro 4.4. Tareas relativas al desarrollo de la investigación (II)

ID	Nombre	Estimación (h)	Fecha fin (dd-mm-yy)	Predecesor
2.3	**Búsqueda de patrones de edición**	**134,4**	**28-02-22**	2.2
2.3.1	Inspección manual de los datos	24	19-01-22	-
2.3.2	Aplicación de métodos de clústering	33,2	14-02-22	2.3.1
2.3.3	Selección de técnicas adicionales de análisis de grafos	6,8	21-01-22	-

Cuadro 4.4. Tareas relativas al desarrollo de la investigación (II)

ID	Nombre	Estimación (h)	Fecha fin (dd-mm-yy)	Predecesor
2.3.4	Aplicación de técnicas adicionales	32,4	02-02-22	2.3.3
2.3.5	Establecimiento de posibles taxonomías con los resultados obtenidos	17,6	21-02-22	2.3.2 2.3.4
2.3.6	Comparación de resultados entre cada fuente de datos	20,8	28-02-22	2.3.5
2.4	**Uso de historial de ediciones en completado de grafos**	**136,0**	**08-04-22**	2.3
2.4.1	Estudio de opciones disponibles a partir del análisis de patrones	33,6	08-03-22	-
2.4.2	Aplicación de técnicas basadas en heurísticos	40,8	18-03-22	2.4.1
2.4.3	Aplicación de técnicas basadas en redes neuronales	61,6	06-04-22	2.4.1
2.4.4	**Evaluación de técnicas implementadas**	**21,2**	**08-04-22**	2.4.2 2.4.3

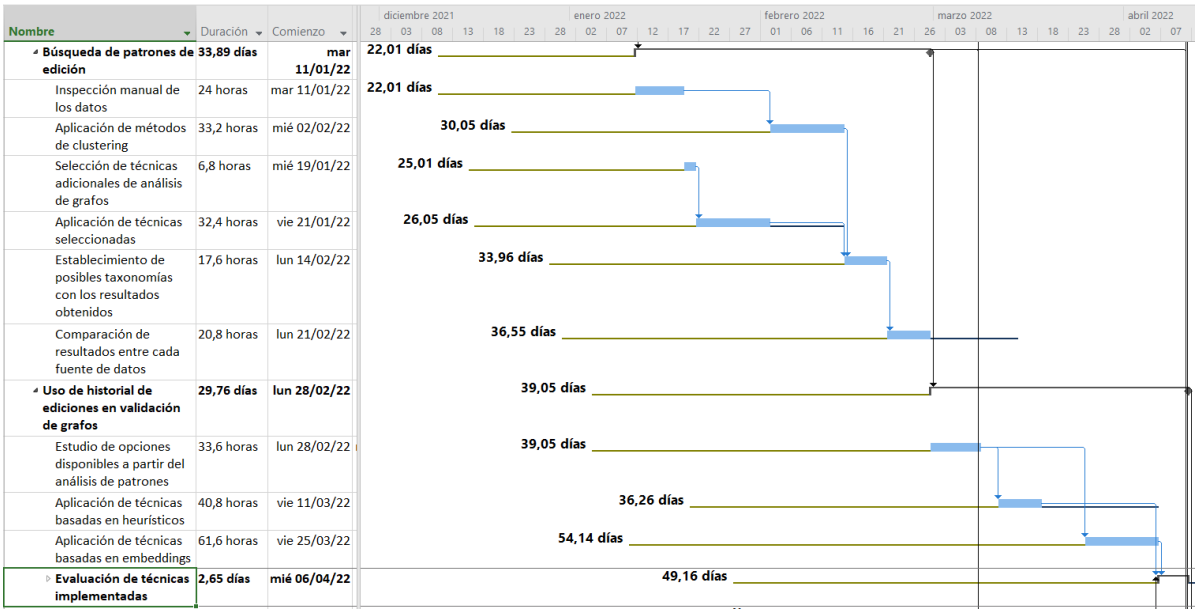


Figura 4.5. Diagrama de Gantt de las tareas relativas al desarrollo de la investigación (II)

Para acabar con el desarrollo de la investigación se muestran en la Tabla 4.5 las tareas de análisis y difusión de los resultados obtenidos. Se estima ejecutar estas tareas entre el **11 de abril** y el **18 de mayo**, con una duración de **160.7 horas**.

Cuadro 4.5. Tareas relativas al desarrollo de la investigación (III)

ID	Nombre	Estimación (h)	Fecha fin (dd-mm-yy)	Predecesor
2.5	**Análisis final de resultados**	**14,8**	**12-04-22**	2.4
2.5.1	Respuesta a las preguntas de investigación planteadas	6,0	**11-04-22**	-
2.5.2	Definición de áreas de mejora y trabajo futuro	8,8	12-04-22	-
2.6	**Implementación de prototipo de validador**	**14,8**	**19-04-22**	2.4
2.6.1	Carga de modelos creados	5,2	18-04-22	-
2.6.2	Implementación funcionalidad de validación	8,8	13-04-22	-
2.6.3	Documentación de instalación y uso	4,2	19-04-22	-
2.6.4	Pruebas de carga y rendimiento	8,4	14-04-22	-
2.6.5	Pruebas metamórficas del validador	9,2	18-04-22	-
2.7	**Difusión de resultados**	**145,9**	**18-05-22**	2.5 2.6
2.7.1	**Publicación de código fuente**	**34,4**	**18-05-22**	-
2.7.2	**Envío de artículo**	**68,3**	10-05-22	-
2.7.3	**Publicación de materiales complementarios**	**43,2**	17-05-22	-

La siguiente tanda de tareas se corresponde con aquellas relativas a la monitorización del proyecto.

Cuadro 4.6. Tareas relativas a la monitorización del proyecto

ID	Nombre	Estimación (h)	Fecha fin (dd-mm-yy)	Predecesor
3	**Monitorización del proyecto**	**61.4**	**25-03-22**	-
3.1	Monitorización del estado de la planificación	14,4	24-03-22	-
3.2	Monitorización y actualización de riesgos del proyecto	20	11-03-22	-

Cuadro 4.6. Tareas relativas a la monitorización del proyecto

ID	Nombre	Estimación (h)	Fecha fin (dd-mm-yy)	Predecesor
3.3	Monitorización y actualización del presupuesto del proyecto	10,8	25-03-22	-
3.4	Mantenimiento y actualización de la bitácora del proyecto	16,2	22-03-22	-

Por último tenemos las tareas relativas al cierre del proyecto. Esta parte incluye el análisis a posteriori del rendimiento en el proyecto, la finalización y revisión de la memoria, y la creación de diapositivas para la defensa del proyecto. Estas tareas están estimadas desde el **12 de mayo** hasta el **25 de mayo**.

Cuadro 4.7. Tareas relativas al cierre del proyecto

ID	Nombre	Estimación (h)	Fecha fin (dd-mm-yy)	Predecesor
4	**Cierre del proyecto**	**135,5**	**25-05-22**	2, 3
4.1	**Análisis de desempeño en el proyecto**	**25,9**	**23-05-22**	-
4.1.1	Creación de planificación final	6,8	19-05-22	-
4.1.2	Creación de presupuesto final	9,2	20-05-22	4.1.1
4.1.3	Análisis de planificación final	3,3	20-05-22	4.1.1
4.1.4	Análisis de presupuesto final	4,4	23-05-22	4.1.2
4.1.5	Documentación de aspectos positivos y negativos de la gestión y desempeño en el proyecto	2,2	23-05-22	4.1.4 4.1.5
4.2	Ajustes a la memoria antes de la entrega	16.8	25-05-22	4.1
5	Reuniones con director (8)	8	02-05-22	

Estimación de tiempos

Todas las tareas que hemos mostrado previamente tienen una duración estimada en horas. El valor asignado se obtuvo utilizando la técnica de PERT modificada. Esta técnica consiste en estimar el tiempo optimista (O) que se tarda en completar una tarea, el tiempo medio

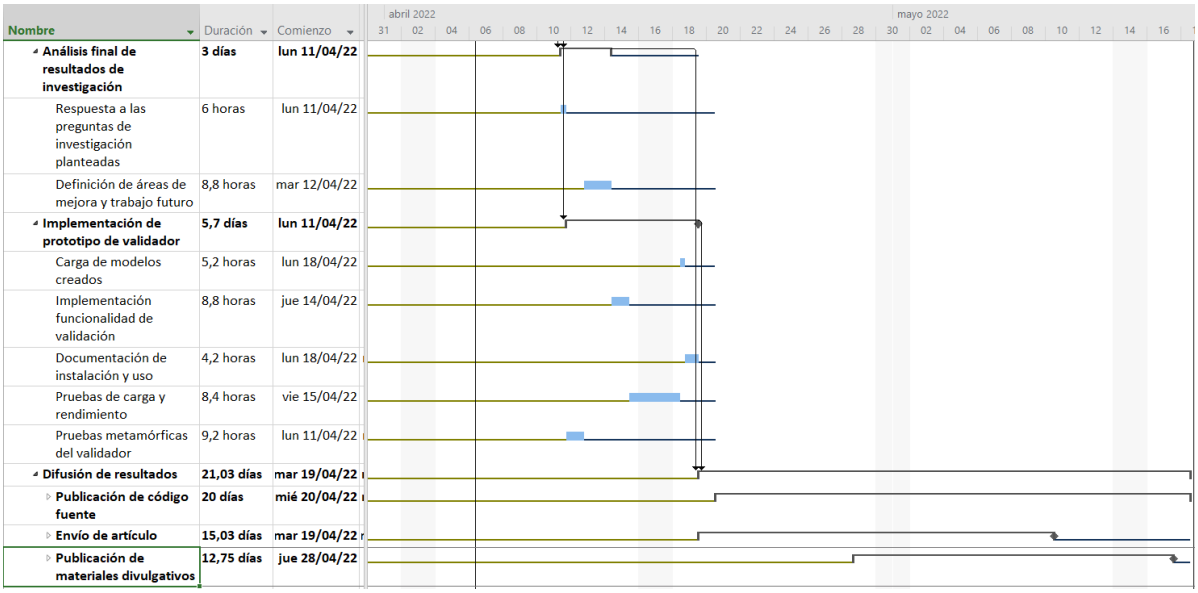


Figura 4.6. Diagrama de Gantt de las tareas relativas al desarrollo de la investigación (III)

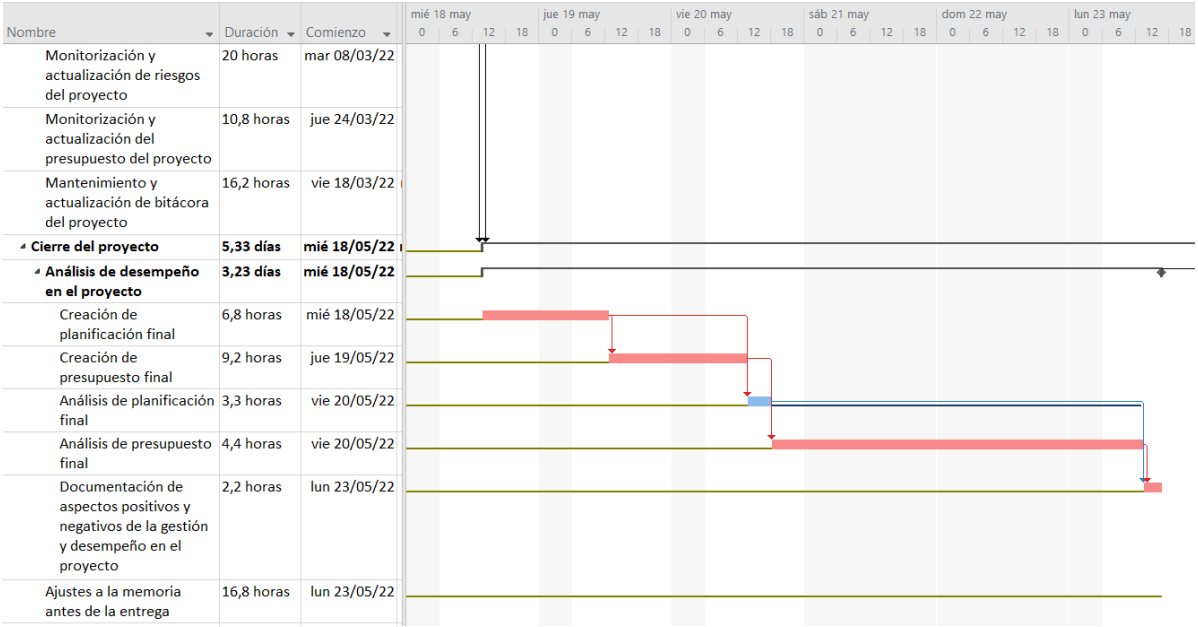


Figura 4.7. Diagrama de Gantt de las tareas relativas al cierre del proyecto

o esperado (M) y el tiempo pesimista (P). Estos tiempos se ponderan posteriormente de la siguiente forma para obtener la estimación final de la tarea:

$$E_{pert} = \frac{O + \gamma * M + P}{\gamma + 2} \tag{4.1}$$

El valor de γ se define con el fin de dar un mayor o menor peso a la estimación media. En la técnica PERT γ es una constante con valor 4, pero con PERT modificado se puede ajustar su valor. En nuestro caso trabajamos con un valor de $\gamma = \mathbf{3}$, en el que asignamos un poco más de valor a los casos optimistas y pesimistas que por defecto.

El uso de la técnica PERT permite tener en cuenta escenarios optimistas y pesimistas en una tarea que tienden a ajustar mejor su estimación final, pero también nos permite hacer un análisis más detallado del proyecto. La fórmula de PERT sigue aproximadamente una

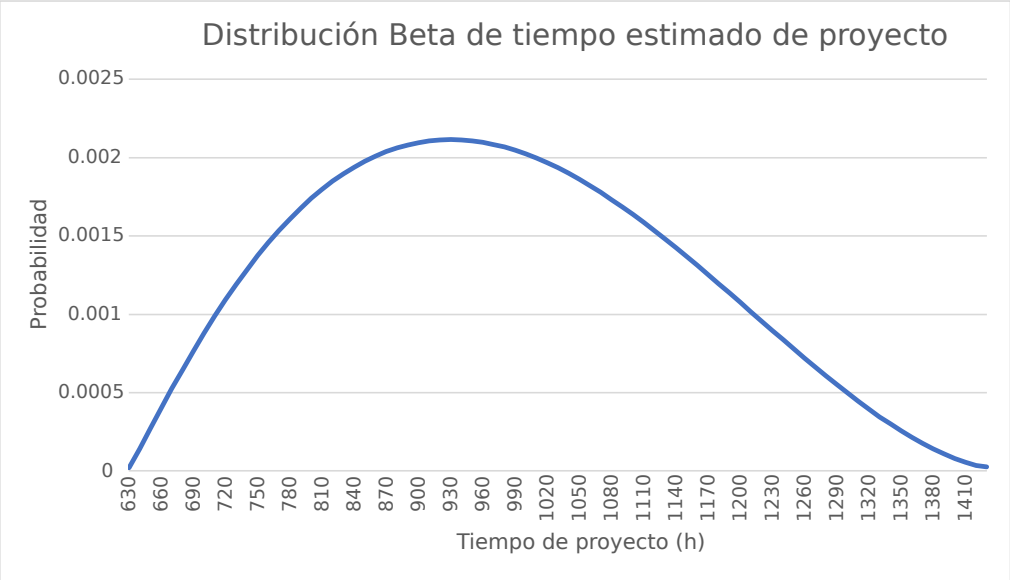


Figura 4.8. Distribución Beta construida a partir de las estimaciones

Cuadro 4.8. Probabilidades de haber finalizado el proyecto en cada fecha tomando como referencia las estimaciones realizadas

Fecha fin (dd-mm-yyyy)	Probabilidad (%)
01-04-2022	15,44
04-05-2022	51,67
03-06-2022	83,19
08-07-2022	99,07

distribución beta, lo cual nos proporciona varias herramientas para analizar las estimaciones realizadas con métodos estadísticos.

A efectos prácticos, con esta distribución beta podemos calcular la probabilidad de que el proyecto se complete dentro de un rango de tiempo determinado dadas las estimaciones que hemos realizado. En la tabla 4.8 se muestran una serie de intervalos de tiempo del proyecto, junto con la probabilidad de que el proyecto se cumpla en ese intervalo[2].

Resumen de la planificación

El resumen del proyecto en el plano temporal se muestra en la Tabla 4.9.

4.1.4. Gestión de riesgos

Plan de gestión de riesgos

En el apéndice B Plan de Gestión de Riesgos se adjunta el plan de riesgos definido para el proyecto.

[2]Es importante indicar que todos los cálculos realizados dependen de la **calidad de las estimaciones** asignadas a cada tarea del proyecto. Aún así, nos sirven como una guía para poder medir la probabilidad de cumplir los tiempos objetivo del proyecto teniendo en cuenta los mejores y peores casos que hemos estimado.

Cuadro 4.9. Resumen de la planificación del proyecto

Número de tareas:	88
Duración de tarea mayor:	61,6h
Duración de tarea menor:	1h
Número de tareas críticas:	8
Media de holguras	21,34h
Fecha de inicio:	01-09-2021
Fecha de fin:	25-05-2022
Duración total del proyecto	190,68 días
Horas totales de trabajo:	975,6 h

Registro de riesgos

En la Tabla 4.10 se muestran los riesgos identificados en el proyecto, junto con su probabilidad e impacto estimados, y la puntuación final en base al plan de gestión de riesgos definido.

Posteriormente se plantearon respuestas a los riesgos identificados. Para decidir la respuesta que aplicar a cada riesgo tuvimos en cuenta su **puntuación** asignada y la **viabilidad** de poder aplicar cada respuesta. También tuvimos en cuenta el propio carácter académico del proyecto a la hora de considerar opciones. Por ejemplo, algunas respuestas posibles como transferir un riesgo a través de subcontrataciones son prácticamente imposibles teniendo en cuenta la naturaleza de este proyecto. Las respuestas propuestas para cada riesgo se presentan en la Tabla 4.11.

Cuadro 4.10. Lista de riesgos identificados

ID	Nombre	Descripción	Probabilidad	Impacto				Puntuación
				Presupuesto	Tiempo	Alcance	Calidad	
R1	Planificación pesimista	Se sobreestima el tiempo de las tareas que componen el proyecto	Baja	Bajo	Medio	Inapreciable	Inapreciable	**0,090**
R2	Recursos computacionales insuficientes	Al trabajar con una cantidad de datos muy grande es posible que las estimaciones de recursos no sean suficientes para llevar a cabo algunas tareas.	Alta	Alto	Medio	Alto	Medio	**0,385**
R3	Pérdida de datos	Una pérdida de los datos extraídos durante el proyecto podría conllevar el tener que repetir el proceso de recogida y limpieza de datos por completo.	Media	Alto	Crítico	Bajo	Bajo	**0,450**
R4	Modificaciones en el alcance del proyecto	Se descubre nueva información durante el desarrollo del proyecto que abre las puertas a distintas tareas y experimentos.	Alta	Medio	Alto	Alto	Medio	**0,385**
R5	Accidente/ Enfermedad	Dado que sólo se cuenta con una persona trabajando en el proyecto, un accidente o enfermedad pararía por completo el progreso en el proyecto hasta que se encontrara en condiciones de retomar el trabajo.	Baja	Medio	Crítico	Medio	Medio	**0,270**

Cuadro 4.10. Lista de riesgos identificados

ID	Nombre	Descripción	Probabilidad	Impacto				Puntuación
				Presupuesto	Tiempo	Alcance	Calidad	
R6	Planificación optimista	Se subestima el tiempo de las tareas del proyecto.	Alta	Medio	Alto	Medio	Alto	**0,385**
R7	No cumplir expectativas	Es posible que se llegue a la fecha fin del proyecto sin haber cumplido algunos de los objetivos fijados o habiéndose desviado de los objetivos iniciales.	Media	Medio	Alto	Alto	Crítico	**0,450**
R8	Aparece publicación similar durante el tiempo de investigación	Mientras se lleva a cabo el proyecto aparece en paralelo una investigación que trata el mismo tema, pudiendo solapar parte del trabajo que ya ha sido realizado.	Baja	Inapreciable	Bajo	Bajo	Alto	**0,165**
R9	Incumplir licencias de librerías utilizadas en el proyecto	A lo largo del proyecto se harán uso de librerías externas. Un incumplimiento de las licencias de éstas podría conllevar acciones legales.	Media	Alto	Medio	Bajo	Bajo	**0,275**

Cuadro 4.10. Lista de riesgos identificados

ID	Nombre	Descripción	Probabilidad	Impacto				Puntuación
				Presupuesto	Tiempo	Alcance	Calidad	
R10	Los datos extraídos no son apropiados para las siguientes fases	Es posible que los datos extraídos al comienzo del proyecto no sigan la estructura que se espera (p.ej., más cambios debido a vandalismo). En este caso las siguientes fases de la investigación podrían verse afectadas.	Media	Bajo	Bajo	Bajo	Medio	**0,150**
R11	Cambios en la API de Wikipedia y/o Wikidata	Un cambio en las llamadas necesarias para obtener datos, así como una política más restrictiva en el número de llamadas que se pueden hacer por minuto podrían retrasar el proyecto de manera significativa.	Baja	Medio	Alto	Bajo	Bajo	**0,165**
R12	Rechazo de autorización de defensa	Para poder presentar el TFM es necesario recibir una autorización de defensa por la comisión académica. En caso de cometer algún error en esta solicitud, no será posible presentar el trabajo.	Baja	Bajo	Alto	Bajo	Bajo	**0,165**

Cuadro 4.10. Lista de riesgos identificados

ID	Nombre	Descripción	Probabilidad	Impacto				Puntuación
				Presupuesto	Tiempo	Alcance	Calidad	
R13	Retraso en el envío del artículo	Dado que el alumno nunca ha realizado el proceso de envío de artículo es posible que surjan imprevistos a la hora de enviar el artículo (p.ej., necesidad de adjuntar documentos adicionales), conllevando un retraso en el envío del mismo.	Media	Bajo	Alto	Bajo	Bajo	**0,275**
R14	El uso del historial de ediciones no mejora los sistemas actuales	Es posible que la principal hipótesis sobre la que se basa el trabajo no se cumpla, disminuyendo considerablemente la calidad de la investigación realizada (especialmente de los resultados).	Alta	Bajo	Bajo	Bajo	Medio	**0,210**
R15	Falta de documentación en las librerías de grafos de conocimiento utilizadas	Las librerías de grafos de conocimiento son relativamente novedosas y escasas, por lo que es posible que exista una falta de documentación (especialmente en las técnicas más actuales) que retrase el desarrollo o genere problemas adicionales.	Media	Medio	Medio	Bajo	Bajo	**0,150**

Cuadro 4.11. Respuestas planteadas para cada riesgo

ID	Nombre	Tipo de respuesta	Respuesta
R1	Planificación pesimista	Asumir el riesgo	Directamente no realizamos ninguna respuesta sobre este riesgo, pero indirectamente se puede ver minimizado por las medidas aplicadas al riesgo R6.
R2	Recursos computacionales insuficientes	Mitigar el riesgo	Se reservará una parte del presupuesto como contingencia en caso de que este riesgo se acabe produciendo, con el fin de utilizar máquinas en la nube para ejecutar las computaciones necesarias.
R3	Pérdida de datos	Mitigar el riesgo	Se utilizará algún sistema de almacenamiento de datos en la nube para mantener una copia de los datos recogidos para la investigación. Siempre que las licencias lo permitan, se facilitará el acceso a estos datos de forma pública a fuentes externas.
R4	Modificaciones en el alcance del proyecto	Mitigar el riesgo	Se establecerán reuniones periódicas (mensualmente) con el director. Una parte de estas reuniones se dedicará a tratar el estado del proyecto y posibles modificaciones a seguir. Toda modificación en los objetivos o tareas definidos en el proyecto deberá ser acordada previamente con el director.
R5	Accidente/ Enfermedad	Asumir el riesgo	Debido a la naturaleza del proyecto, no es posible tratar este riesgo con algunas medidas tradicionales (por ejemplo, contratando personal). Por lo tanto se asume el riesgo, aceptando posibles retrasos en el proyecto si se dan las circunstancias.
R6	Planificación optimista	Mitigar el riesgo	El objetivo es minimizar este riesgo a través de la gestión del tiempo realizada. En primer lugar, la propia estimación de tiempos utilizando PERT tiene en cuenta escenarios optimistas y pesimistas que estabilizan más las estimaciones. En segundo lugar, se realizará un análisis de desempeño en cada hito con el fin de detectar problemas de planificación y corregirlos lo antes posible.
R7	No cumplir expectativas	Mitigar el riesgo	Al estar trabajando en un ámbito académico es posible extender el plazo de entrega si no se cumple la calidad y objetivos acordados con el director. La comunicación también es importante para evitar este problema, o al menos detectarlo lo antes posible. Las reuniones mensuales planteadas también buscan minimizar este problema. En cada reunión se tratará el avance y situación del proyecto hasta la fecha.

Cuadro 4.11. Respuestas planteadas para cada riesgo

ID	Nombre	Tipo de respuesta	Respuesta
R8	Aparece publicación similar durante el tiempo de investigación	Asumir el riesgo	La aparición de publicaciones similares durante el proyecto es algo que está fuera de nuestro control, por lo que es necesario asumir el riesgo. Dado que el proyecto dura unos 9 meses, se añadirá una tarea en el cierre del proyecto de repasar la literatura que ha surgido durante el proyecto y actualizar las partes de la memoria afectadas con esta nueva información.
R9	Incumplir licencias de librerías utilizadas en el proyecto	Mitigar el riesgo	Con el fin de evitar problemas de incumplimientos de licencias se mantendrá un documento con todas las librerías externas utilizadas. Para cada librería se anotará, como mínimo, lo siguiente: nombre, breve descripción, para qué se utiliza, y la licencia que tiene.
R10	Los datos extraídos no son apropiados para las siguientes fases	Asumir el riesgo	La propia investigación busca en sí el determinar hasta que punto los datos del historial de ediciones de un grafo son reaprovechables. Se considera que este es un riesgo inherente a la investigación realizada, por lo que se acepta.
R11	Cambios en la API de Wikipedia y/o Wikidata	Asumir el riesgo	Ya que no tenemos control sobre las modificaciones que haga Wikipedia o Wikidata sobre sus APIs, aceptamos el riesgo y las posibles alteraciones a la planificación que ocurran como consecuencia.
R12	Rechazo de autorización de defensa	Asumir el riesgo	Se asume el riesgo, teniendo en cuenta que en caso de que ocurra se podría preparar la solución de cara a la siguiente convocatoria.
R13	Retraso en el envío del artículo	Mitigar el riesgo	Se ha reservado un colchón de tiempo en la planificación para poder gestionar imprevistos como este.
R14	El uso del historial de ediciones no mejora los sistemas actuales	Asumir el riesgo	Teniendo en cuenta el contexto del trabajo asumimos el riesgo. En caso de que no se cumplan las hipótesis planteadas seguiremos adelante con el trabajo y la redacción del artículo.
R15	Falta de documentación en las librerías de grafos de conocimiento utilizadas	Mitigar el riesgo	Las tareas de desarrollo de la investigación en las que es necesario utilizar estas librerías se han estimado con un extra de tiempo para tener en cuenta este posible riesgo.

Cuadro 4.12. Presupuesto inicial de cliente (resumido)

Cod.	Partida	Total
1	Estudio de validación de grafos de conocimiento utilizando el historial de ediciones	42.658,75€
2	Implementación de prototipo de validador	6.424,51€
3	Creación de materiales divulgativos	12.450,44€
	Subtotal	**61.532,69€**
	IVA (21 %)	**12.921,86€**
	TOTAL CLIENTE:	**74.454,55€**

4.1.5. Presupuesto inicial

En la Tabla 4.12 se muestra el presupuesto de cliente resumido para este proyecto. En el apéndice D Presupuesto Detallado se detalla como se ha llegado a este presupuesto, y se desarrollan las partidas que lo componen.

4.2. Ejecución del proyecto

4.2.1. Plan seguimiento de planificación

Para realizar un seguimiento del desarrollo del proyecto se establecieron las siguientes 5 líneas base, que coinciden con puntos clave del mismo:

- **1 de septiembre de 2021**: Esta es la línea base inicial, en la que se comienza el desarrollo del proyecto.
- **15 de enero de 2022**: Este es el punto en el que se tenía pensado haber acabado las tareas iniciales de gestión del proyecto, la revisión de literatura, y obtenido los datos con los que realizar los experimentos.
- **15 de marzo de 2022**: Este punto se consideró bastante importante ya que a partir de esa fecha aproximadamente se finalizan las asignaturas del máster, pasando el proyecto de tener un calendario a media jornada a tener uno de jornada completa. Aunque a efectos de días se encuentra en un estado avanzado de proyecto (pasaron unos 6 meses desde el comienzo del proyecto y quedan unos 3), en cuanto a horas de trabajo se encuentra en un punto prácticamente intermedio.
- **02 de mayo de 2022**: Este punto se estableció para poder analizar el desempeño del proyecto antes del último mes y poder tomar las últimas medidas que se consideraran necesarias.
- **06 de junio de 2022**: Este punto se corresponde con el fin del proyecto.

En cada una de estas líneas base se calculó el valor planificado para ese instante de tiempo, el valor real que se consiguió, y los costes que se asumieron para conseguir ese valor. A continuación pasamos a analizar el desempeño durante cada uno de estos puntos temporales, acompañándonos de una curva S creada a partir del análisis del método del valor ganado realizado. Esta curva S se muestra en la Figura 4.9.

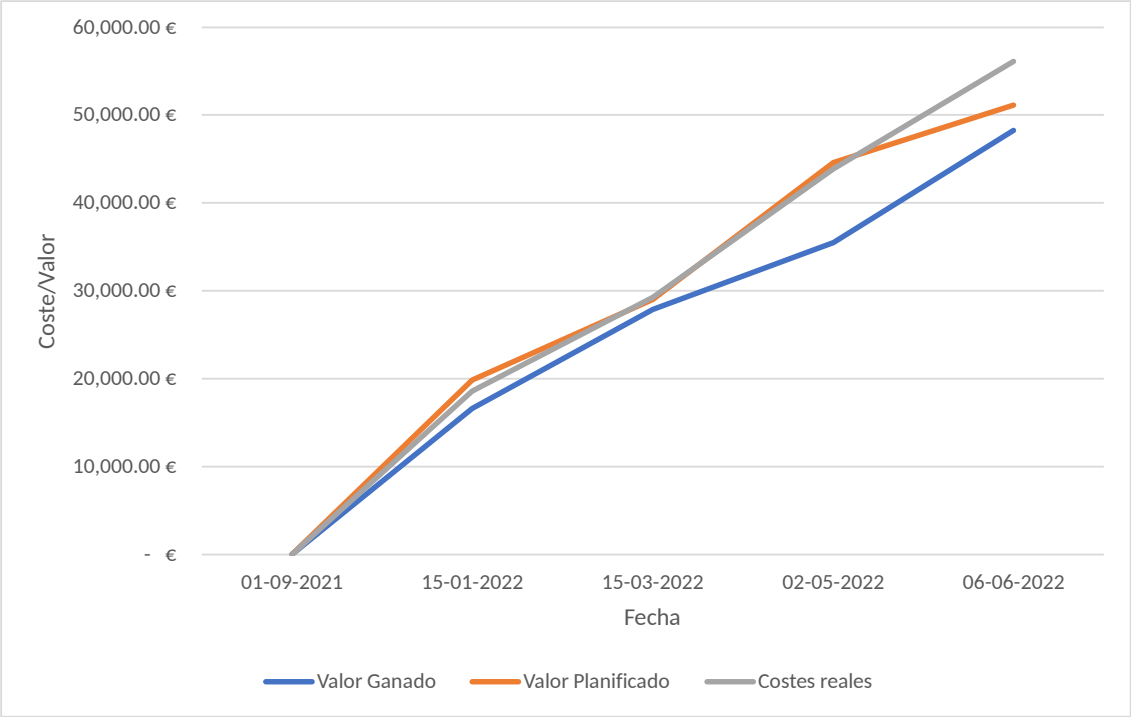


Figura 4.9. Curva S obtenida

Durante el primer tramo de trabajo (hasta el 15 de enero de 2022) se alargaron algunas tareas iniciales de gestión de proyecto más de lo esperado[3], y además se produjeron problemas al obtener los datos ya que el volumen de datos era mucho mayor de lo esperado, teniéndole que dedicar más tiempo del planificado. Esto hizo que aunque los costes reales sean prácticamente idénticos al valor planificado, el valor realmente ganado fuera bastante menor que el previsto a estas alturas.

En este punto ya se detectó que las cosas no iban del todo bien, ya que el hecho de trabajar con un volumen de datos tan grande no sólo había retrasado el desarrollo del proyecto hasta el momento sino que también retrasaría alguna de las tareas posteriores que trabajaban sobre esos datos. En una empresa real podríamos intentar contratar más personal o aumentar la productividad del personal existente para intentar generar todo el valor planificado, aunque conlleve un aumento de los costes. De cara a nuestro trabajo optamos por otra decisión distinta: eliminar tareas que no fueran críticas para el desarrollo del proyecto si la situación lo requería. Esta decisión se tomó porque las horas planificadas para el proyecto ya se encontraban muy por encima de los requisitos para el trabajo Fin de Máster en cuanto a horas por crédito se refiere, y aumentar todavía más estas horas para poder cumplir todas las tareas previstas era inasumible.

El 15 de marzo de 2022 se volvió a analizar la situación, y la cosa se había estabilizado bastante. Se habían compensado algunas horas que no se habían realizado en el primer tramo, y algunas tareas no relacionadas con el manejo de datos tardaron menos de lo esperado, pudiendo recuperar trabajo retrasado.

De cara a la siguiente línea base, del 2 de mayo de 2022, la situación empeoró drásticamente. Esto se debe a que en ese momento se ejecutaron los modelos de aprendizaje automático que trabajaban sobre los datos obtenidos previamente, y los tiempos de entrenamiento y de evaluación superaron con creces lo esperado. Esto se unió a que la redacción del artículo también llevó más de lo esperado, haciendo que en ese punto tuvieramos un retraso considerable con el

[3]En la Subsección 4.3.1 - Planificación final se mostrarán de forma concreta los retrasos y adelantos producidos durante el proyecto que mencionamos en este análisis.

valor planificado.

Las soluciones que se tomaron fueron las siguientes: se eliminaron algunas de las tareas restantes (p.ej., la creación de la demo online y pruebas con modelos basados en heurísticos), y se retrasó la fecha de finalización del proyecto 10 días más (hasta el 6 de junio).

Cuadro 4.13. Bitácora de incidencias del proyecto

Fecha	Descripción	Decisiones
05-10-21	Se detectó la necesidad de realizar una revisión de literatura sobre estudios de calidad en grafos de conocimiento, que no había sido contemplada en la planificación inicial.	Dado que el proyecto empezó hace poco, y que sigue los tiempos definidos por el momento, se añade la nueva tarea a la planificación.
19-11-21	Los métodos de obtención de datos de ediciones planeados no pueden ser utilizados, por lo que es necesario dedicar más tiempo a la obtención de datos.	De momento se mantienen todas las tareas y tiempos planificados, asumiendo este retraso.
20-01-22	El tamaño de los datos es mucho mayor de lo esperado, ralentizando el desarrollo de los análisis sobre éstos.	Se eliminan tareas de clusterización para compensar por el tiempo de análisis extra del resto de tareas.
27-03-22	No se tienen recursos computacionales (en este caso, RAM) necesarios para entrenar algunos de los modelos de *embeddings* de grafos de conocimiento.	Se opta por utilizar otros modelos alternativos, que consumen menos recursos y siguen teniendo resultados competitivos según varios benchmarks existentes.
10-04-22	Los tiempos de entrenamiento y de evaluación de los modelos son extremadamente más lentos de lo planificado	Se utiliza el presupuesto reservado para el riesgo RI2 para adquirir una máquina en la nube con la que el entrenamiento y la evaluación vayan más rápidos, cumpliendo con los tiempos definidos.
02-05-22	La escritura y corrección del artículo lleva más tiempo del esperado.	Se elimina la tarea de publicar una demostración online de los sistemas y se utilizan 10 días extra tras la fecha de fin de proyecto planificada para finalizar todas las tareas retrasadas por este problema.

El resultado final detecta algunos aspectos del proyecto que no fueron todo lo bien que deberían: se dedicaron más horas de las planificadas inicialmente, y a pesar de ello todavía no se consiguió todo el valor planificado inicialmente. Aún así, dentro de un proyecto real se seguirían teniendo beneficios si tenemos en cuenta los planes de contingencia ante imprevistos definidos. El mayor problema en ese contexto sería que no todo el valor esperado por el cliente pudo ser cumplido. Una posible solución podría ser la continuación del proyecto hasta cumplir todo ese valor, aunque los costes aumentaran. Teniendo en cuenta nuestro contexto real de trabajo académico, en general estamos satisfechos con el desempeño realizado y las decisiones tomadas, aunque hay varios aspectos a mejorar que se mencionan en la Subsección 4.3.4 - Informe de lecciones aprendidas.

4.2.2. Bitácora de incidencias del proyecto

En la Tabla 4.13 se muestra la bitácora de incidencias registradas durante el proyecto.

4.2.3. Riesgos

En el apéndice C - Hojas de riesgos se muestran las hojas de cada riesgo priorizado previamente. Para cada uno de estos riesgos se realiza la monitorización.

4.3. Cierre del proyecto

4.3.1. Planificación final

A continuación pasamos a mostrar la planificación final tras acabar el proyecto. A lo largo del transcurso del proyecto se tuvieron que añadir o eliminar tareas que no estaban previstas en un principio, así como modificar los tiempos estimados en muchos casos. Agruparemos las tareas finales de la misma forma que con la planificación inicial, marcando con colores aquellos casos en los que la tarea sea nueva o haya sufrido modificaciones en su tiempo final.

La primera tanda de tareas son aquellas relativas al inicio del proyecto, y se muestran en la Tabla 4.14. Por lo general no hubo demasiados cambios en estas tareas, a salvo de algunos casos en los que la duración real de la tarea fue algo superior a lo estimado inicialmente.

Cuadro 4.14. Tareas finales relativas al inicio del proyecto

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
1	**Inicio del proyecto**	**94,2 (+4,8)**	**11-10-21**	-
1.1	Definición de objetivos y alcance del proyecto	8,8	02-09-21	-
1.2	Creación de acta de proyecto	8,8	22-09-21	-
1.3	Desglose en tareas	12,4	03-09-21	1.1
1.4	Estimar duraciones de las tareas	10,4	07-09-21	1.3
1.5	Estructura inicial de la memoria	8,2	24-09-21	-
1.6	**Gestión de riesgos**	**18,8**	**01-10-21**	1.1
1.6.1	Definición de lista de riesgos	12,2	30-09-21	-
1.6.2	Estudio de medidas a adoptar	**6 (+1,6)**	30-09-21	1.6.1
1.6.3	Refinar tareas con las medidas adoptadas	2,2	01-10-21	1.6.2
1.7	**Creación de presupuesto inicial**	**22**	**11-10-21**	1.4
1.7.1	Cálculo de costes del proyecto	**8 (+3,2)**	06-10-21	-
1.7.2	Creación de presupuesto interno	12,8	08-10-21	-

Cuadro 4.14. Tareas finales relativas al inicio del proyecto

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
1.7.3	Creación de presupuesto externo (cliente)	4,4	11-10-21	-

En la Tabla 4.15 se muestran las tareas finales relativas a la primera fase de la investigación. Podemos ver como aquí ya surgió el primer problema grave en cuanto a la ejecución del proyecto que ya se comentó previamente: las fuentes de datos que teníamos pensado utilizar no estaban accesibles, y además la cantidad de datos fue mayor de lo esperado. Esto produjo unos aumentos de tiempo considerables en algunas tareas.

Cuadro 4.15. Tareas finales relativas al desarrollo de la investigación (I)

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
2	**Desarrollo de la investigación**	**881,9 (+139,2)**	31-05-22	1.1, 1.3, 1.4
2.1	**Revisión de literatura y estado del arte**	**136,0 (+6,8)**	09-11-21	-
2.1.1	Uso del historial de revisiones en grafos de conocimiento	**16,0 (-8,4)**	15-10-21	-
2.1.2	Algoritmos de corrección de grafos de conocimiento	**52,0 (+4,0)**	21-09-21	-
2.1.3	**Calidad en grafos de conocimiento**	**26,0 (+26,0)**	**09-11-21**	-
2.1.4	Dinámicas de conflicto y colaboración en Wikipedia y Wikidata	**12,0 (-20,4)**	28-10-21	-
2.1.5	Métodos de análisis de grafos	**30,0 (+5,6)**	08-11-21	-
2.2	**Obtención y tratamiento de datos**	**214,0 (+62,0)**	03-02-22	2.1
2.2.1	Selección de subconjunto de datos a utilizar	25,2	17-11-21	-
2.2.2	Obtención de subconjunto de datos	**96,0 (+53,2)**	20-12-21	2.2.1
2.2.3	Filtrado de datos (p.ej., vandalismo)	28,0	12-01-22	2.2.2
2.2.4	Preprocesamiento	12,8	17-01-22	2.2.3
2.2.5	Transformación de datos no estructurados a tripletas	**20,0 (-6,4)**	24-01-22	2.2.4

Cuadro 4.15. Tareas finales relativas al desarrollo de la investigación (I)

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
2.2.6	Creación de grafos de ediciones con los datos obtenidos	**32,0 (+15,2)**	03-02-22	2.2.5

En la Tabla 4.16 se muestran las tareas finales relativas a la segunda fase de la investigación. En esta fase se eliminaron algunas tareas no críticas con el fin de solucionar los problemas de retrasos que estábamos experimentando, como ya hemos comentado previamente en el seguimiento de la planificación.

Cuadro 4.16. Tareas finales relativas al desarrollo de la investigación (II)

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
2.3	**Búsqueda de patrones de edición**	**111,6 (-22,8)**	**10-03-22**	2.2
2.3.1	Inspección manual de los datos	24	09-03-22	-
~~2.3.2~~	~~Aplicación de métodos de clústering~~	~~33,2~~	~~14-02-22~~	~~2.3.1~~
2.3.3	Selección de técnicas adicionales de análisis de grafos	**8,0 (+1,2)**	07-02-22	-
2.3.4	Aplicación de técnicas adicionales	**56,0 (+23,6)**	25-02-22	2.3.3
~~2.3.5~~	~~Establecimiento de posibles taxonomías con los resultados obtenidos~~	~~17,6~~	~~21-02-22~~	~~2.3.2~~ ~~2.3.4~~
2.3.6	Comparación de resultados entre cada fuente de datos	**24,0 (+3,2)**	10-03-22	2.3.5
2.4	**Uso de historial de ediciones en completado de grafos**	**154,8 (+18,8)**	**19-04-22**	2.3
2.4.1	Estudio de opciones disponibles a partir del análisis de patrones	**30,0 (-3,6)**	16-03-22	-
~~2.4.2~~	~~Aplicación de técnicas basadas en heurísticos~~	~~40,8~~	~~18-03-22~~	~~2.4.1~~
2.4.3	Aplicación de técnicas basadas en redes neuronales	**86,0 (+24,4)**	11-04-22	2.4.1

Cuadro 4.16. Tareas finales relativas al desarrollo de la investigación (II)

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
2.4.4	**Evaluación de técnicas implementadas**	**50,0 (+38,8)**	**19-04-22**	2.4.3

En la Tabla 4.17 se muestran las tareas finales relativas a la última fase de la investigación. Durante esta fase se produjo una gran desviación en la redacción y correcciones del artículo respecto a lo planificado.

Cuadro 4.17. Tareas finales relativas al desarrollo de la investigación (III)

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
2.5	**Análisis final de resultados**	**11,8 (-3,0)**	**21-04-22**	2.4
2.5.1	Respuesta a las preguntas de investigación planteadas	**3,0 (-3,0)**	**19-04-22**	-
2.5.2	Definición de áreas de mejora y trabajo futuro	8,8	21-04-22	-
2.6	**Implementación de prototipo de validador**	**15,8 (+1,0)**	**27-04-22**	2.4
2.6.1	Carga de modelos creados	**6,0 (+0,8)**	25-04-22	-
2.6.2	Implementación funcionalidad de validación	8,8	21-04-22	-
2.6.3	Documentación de instalación y uso	**6,0 (+1,8)**	22-04-22	-
2.6.4	Pruebas de carga y rendimiento	**16,0 (+7,6)**	27-04-22	-
~~2.6.5~~	~~Pruebas metamórficas del validador~~	~~9,2~~	~~18-04-22~~	-
2.7	**Difusión de resultados**	**183,5 (+37,6)**	**31-05-22**	2.5 2.6
2.7.1	**Publicación de código fuente**	**36,0 (+1,6)**	**31-05-22**	-
2.7.2	**Envío de artículo**	**115,5 (+47,2)**	30-05-22	-
2.7.3	**Publicación de materiales complementarios**	**32,0 (-11,2)**	18-05-22	-

En la Tabla 4.18 se muestran las tareas finales relativas a la monitorización del proyecto.

En este caso se consiguió recuperar algo de tiempo perdido ya que las duraciones finales fueron en general menores a las estimadas.

Cuadro 4.18. Tareas finales relativas a la monitorización del proyecto

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
3	**Monitorización del proyecto**	**52,4 (-9,0)**	**25-03-22**	-
3.1	Monitorización del estado de la planificación	14,4	24-03-22	-
3.2	Monitorización y actualización de riesgos del proyecto	**16,0 (-4,0)**	11-03-22	-
3.3	Monitorización y actualización del presupuesto del proyecto	**14,0 (+3,2)**	25-03-22	-
3.4	Mantenimiento y actualización de la bitácora del proyecto	**8,0 (-8,2)**	22-03-22	-

Por último, en la Tabla 4.19 se muestran las tareas finales relativas al cierre del proyecto. En este caso de nuevo se volvieron a retrasar algo las tareas respecto al tiempo estimado, principalmente debido a los ajustes y correcciones que realizar a la memoria.

Cuadro 4.19. Tareas finales relativas al cierre del proyecto

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
4	**Cierre del proyecto**	**171,3 (+35,8)**	**03-06-22**	2, 3
4.1	**Análisis de desempeño en el proyecto**	**25,9**	**26-05-22**	-
4.1.1	Creación de planificación final	6,8	23-05-22	-
4.1.2	Creación de presupuesto final	**12,0 (+2,8)**	25-05-22	4.1.1
4.1.3	Análisis de planificación final	3,3	25-05-22	4.1.1
4.1.4	Análisis de presupuesto final	4,4	26-05-22	4.1.2
4.1.5	Documentación de aspectos positivos y negativos de la gestión y desempeño en el proyecto	**4,0 (+1,8)**	26-05-22	4.1.4 4.1.5

Cuadro 4.20. Resumen de la planificación final del proyecto

Número de tareas:	84
Fecha de inicio:	01-09-2021
Fecha de fin:	06-06-2022
Horas totales de trabajo:	1081,1h (+105,5h)

Cuadro 4.19. Tareas finales relativas al cierre del proyecto

ID	Nombre	Duración (h)	Fecha fin (dd-mm-yy)	Predecesor
4.2	Ajustes a la memoria antes de la entrega	**48,0 (+31,2)**	03-06-22	4.1
5.5	Reuniones con director (8)	8	02-05-22	

En la Tabla 4.20 se muestran las estadísticas finales de la planificación del proyecto.

4.3.2. Informe final de riesgos

En las hojas de riesgos disponibles en el apéndice C se incluye el estado final de los riesgos priorizados.

4.3.3. Presupuesto final de costes

En la Sección D.3 del apéndice D - Presupuesto Detallado se muestra el presupuesto final del proyecto.

4.3.4. Informe de lecciones aprendidas

A continuación se muestran las lecciones aprendidas sobre la gestión del proyecto durante el transcurso del mismo:

- La planificación de algunas tareas relacionadas con la investigación fue bastante mejorable, debido a la falta de experiencia que tenía en este ámbito. Por ejemplo, el tiempo de escritura y corrección del artículo fue mucho mayor del esperado debido a diversos factores no contemplados[4]. De cara al futuro se podrá utilizar esta experiencia para estimar mejor este tipo de tareas.

- Otro aspecto que se subestimó bastante fue el volumen de los datos con los que se trabajaba. Desde un principio se tuvo en cuenta el trabajar con un subconjunto del grafo de conocimiento, pero aún así este subconjunto tuvo mucho más información de la esperada. Esto dificultó muchísimo el avance del proyecto en los tiempos planificados. De cara al futuro quizá sería interesante realizar de forma preliminar unas estimaciones de

[4]Por ejemplo, la revista solicitaba un número de páginas bastante mayor al esperado. El propio proceso de envío a la revista también requirió la redacción de documentos adicionales que no me esperaba.

los volúmenes de datos con los que se trabajarán[5], y ya planificar después de una manera mucho más precisa las tareas que sería viable cumplir en el tiempo definido.

- Durante el proyecto se reforzó bastante la importancia de tener reuniones con el director de proyecto periódicas, siempre que su disponibilidad lo permita. El realizar reuniones mensuales ayudó enormemente a mantener el proyecto activo, aún cuando se compaginaba con el Máster y otras obligaciones, y a poder reajustar el alcance del mismo ante los avances que iban surgiendo.
- Otro factor importante que se enfatizó en este proyecto, sobre todo suponiendo que el proyecto estuviera financiado, es la importancia de reservar gastos del proyecto para contingencias. Al final hay muchas opciones de que surjan imprevistos, a pesar de los esfuerzos en la gestión de riesgos que se realicen. En nuestro caso estos imprevistos vinieron de la cantidad de datos utilizada y de los requisitos computacionales requeridos por los modelos de *embeddings* de grafos. Estas reservas permitirían seguir acabando el proyecto con beneficios a pesar de los problemas surgidos.

Conclusiones

En este capítulo hemos visto las tareas de gestión del proyecto realizado, cubriendo tanto la planificación inicial, como posteriormente el seguimiento y cierre del proyecto. Hemos visto los diversos problemas que han surgido durante el desarrollo del proyecto, motivados principalmente por trabajar con un volumen de datos mayor al esperado, así como por la falta de experiencia en proyectos de carácter investigador. En los siguientes capítulos trataremos directamente sobre los experimentos realizados en la investigación y los resultados obtenidos.

[5] En nuestro caso era difícil estimar la cantidad de datos ya que nunca se había trabajado con el historial de ediciones a esta escala, pero incluso dedicar unos días a obtener una muestra de datos y realizar las estimaciones antes de planificar sería recomendable.

Capítulo 5

Enfoque Experimental

Para poder cumplir con los objetivos de la investigación planteados hemos realizado una serie de tareas que detallaremos en este capítulo.

5.1. Extracción de subconjunto de datos de Wikidata

Debido a la inmensa cantidad de datos almacenados en Wikidata[1], nuestro primer paso fue definir un subconjunto de datos sobre el que trabajar y realizar los procesos experimentales planteados.

Para obtener este subconjunto hemos calculado el índice de ClassRank [41] de las clases existentes en Wikidata, seleccionando las 100 clases con mayor valor de ClassRank. Este índice representa la importancia de una clase dentro de la ontología, calculada a partir de los valores de PageRank [42] de cada una de las instancias de ésta. Por lo tanto, con el índice de ClassRank obtenemos una representación de lo probable que es que un usuario acabe visitando una instancia de esa clase. Consideramos que trabajar sobre estas clases aportaría más valor a los resultados que si éstos fueran obtenidos sobre un subconjunto de datos aleatorio.

Para la ejecución de ClassRank definimos como *class-pointer* la propiedad *P31 (instancia de)* de Wikidata. Como acabamos de comentar, para obtener el valor de ClassRank de una clase es necesario tener el valor de PageRank de cada una de sus instancias. Dado que calcular el valor de PageRank de todas las instancias de Wikidata no era posible con nuestros recursos computacionales, se optó por utilizar unos valores de PageRank precalculados. Estos valores se obtienen a partir de la herramienta Danker[2], que periódicamente calcula los valores de PageRank para cada entidad existente en Wikipedia [43]. Posteriormente, se enlazan las entidades de Wikipedia con su equivalente en Wikidata, obteniendo así una aproximación del valor de PageRank de cada entidad de Wikidata[3].

Las 20 clases más importantes obtenidas según ClassRank se muestran en la tabla 5.1. De los resultados obtenidos por ClassRank se eliminaron manualmente entidades que a nivel ontológico de Wikidata son consideradas como clases, pero que a nivel conceptual no lo serían. Para ello, a modo de filtro básico se eliminaron todas las clases que contenían *"Wikimedia"* en su etiqueta, ya que estas entidades se utilizan para organizar contenido de Wikimedia y no para representar clases a nivel conceptual.

El subconjunto final está compuesto por 89 clases y 9,3 millones de instancias, constituyendo aproximadamente un 10 % de las entidades totales de Wikidata. Es importante tener en

[1] A fecha de redacción de este documento Wikidata cuenta con 71.611.020 entidades, ocupando más de 1TB de almacenamiento (sin incluir datos del historial de ediciones). Teniendo en cuenta el alcance, tiempo, y recursos tecnológicos del proyecto, trabajar sobre el conjunto total de datos sería muy difícil.

[2] https://github.com/athalhammer/danker

[3] Accesibles a través de: https://danker.s3.amazonaws.com/index.html

cuenta que aunque sólo se contenga un 10 % de instancias de Wikidata, el dataset obtenido es aproximadamente un 35 % del tamaño total de Wikidata. Esto se debe a que las instancias que pertenecen a este conjunto de clases importantes tienden a tener mucho más contenido introducido por la comunidad frente a otras instancias, ocupando más tamaño.

Cuadro 5.1. Top 20 clases más importantes de Wikidata según ClassRank

Nombre	Valor Classrank	Número de instancias
human	2.167.439	3.873.812
Wikimedia category	1.057.559	2.207.283
sovereign state	837.883	203
taxon	755.681	1.962.491
country	746.208	193
point in time with respect to recurrent timeframe	635.401	3.273
calendar year	499.551	666
big city	321.893	3.238
human settlement	321.637	512.417
Wikimedia disambiguation page	317.631	1.294.218
Wikimedia administration category	302.196	12.146
Wikimedia list article	287.605	301.370
language	284.433	8.894
modern language	257.324	6.875
city	247.022	8.650
academic discipline	204.426	1.603
time zone named for a UTC offset	170.254	72
metacategory in Wikimedia projects	167.871	2.545
republic	165.881	78
capital	161.384	388
taxonomic rank	160.822	67

5.2. Obtención de datos de revisiones

Con el subconjunto de clases e instancias de nuestros experimentos seleccionado, el siguiente paso fue obtener un dataset con el historial de ediciones completo de cada entidad. Como se ha mencionado en el capítulo 3, ninguno de los proyectos existentes para manejar el historial de ediciones de Wikidata era usable durante el desarrollo del proyecto:

- Wikidated se encuentra en un proceso de desarrollo activo a fecha de escritura de esta memoria, por lo que durante la realización del proyecto todavía no era posible utilizar la herramienta para generar el dataset de revisiones de Wikidata.

- El servicio de consulta del historial de Wikidata producía un error 502 durante la fase de obtención de datos de revisiones del proyecto (Octubre 2021-Febrero 2022), descartándolo como posible alternativa[4]. Además, este servicio cuenta sólo con los datos de revisiones de las entidades hasta 2019, faltando toda la información editada desde entonces en las propiedades.

Esta situación llevó a la obtención de estos datos de revisiones particular para el desarrollo de los experimentos. Los datos crudos de revisiones se obtuvieron a partir de los dumps de Wikidata ofrecidos por Wikimedia[5] a fecha de 1 de noviembre de 2021. Dentro de este repositorio de datos se encuentran unos archivos *XML* con el contenido completo de cada página de Wikidata tras cada edición, comprimidos en formato *bz2*. Este fue el conjunto de datos utilizado para los experimentos, ya que contiene el contenido en JSON de cada entidad de Wikidata tras realizarse cada revisión sobre ésta.

Debido al gran tamaño de este dataset[6], se realizaron una serie de procesamientos especiales para poder trabajar con él de cara al trabajo presentado. En primer lugar, en vez de almacenar el contenido en JSON completo de cada entidad tras cada revisión r, lo que se hizo fue calcular el diff entre el JSON de la entidad tras la revisión actual (r_t) y el JSON tras la siguiente revisión (r_{t+1}). Por lo tanto, si una entidad de Wikidata tiene 10 revisiones, en vez de almacenar el JSON completo de la entidad tras cada revisión lo que se hizo fue almacenar 10 diffs de la entidad (1 por cada revisión). De esta forma, acumulando los n primeros diffs se puede obtener el estado de la entidad tras la revisión número n. En la figura 5.1 se muestra un ejemplo de esta decomposición en diffs.

Nota: JSON Patch Format

Para obtener los diffs entre los documentos JSON de una entidad tras cada revisión se siguió el formato de JSON Patch[a]. Este formato está estandarizado, y define la estructura de un documento JSON para expresar una secuencia de operaciones que se deben aplicar a otro documento JSON. De esta forma, podemos definir una serie de documentos con operaciones que aplicar tras cada revisión a la entidad de Wikidata, obtieniendo el contenido JSON de la entidad tras aplicar la revisión. Existen multitud de librerías en distintos lenguajes que permiten calcular el Patch entre dos documentos JSON o aplicar una serie de Patchs a un documento.

[a] https://datatracker.ietf.org/doc/html/rfc6902

En este proceso se extrajo la siguiente información de cada revisión:

- **id**: Identificador de Wikidata de la revisión.
- **parent_id**: Identificador de Wikidata de la revisión anterior a ésta.
- **entity_id**: Identificador de Wikidata de la entidad a la que la revisión afecta.
- **timestamp**: Fecha de realización de la revisión, siguiendo el formato ISO 8601 (p.ej., +2019-05-27T09:31:10Z).
- **username**: Nombre del usuario que ha realizado la revisión.

[4] Este error ya no se produce a fecha de escritura de esta memoria.
[5] Accesibles a través de: https://dumps.wikimedia.org/wikidatawiki/
[6] En ningún momento se pudo descomprimir el dataset completo debido a falta de espacio en disco duro, pero el tamaño original del dataset sin descomprimir es de 1TB, pudiendo estimar el tamaño del dataset final en decenas de TB al menos.

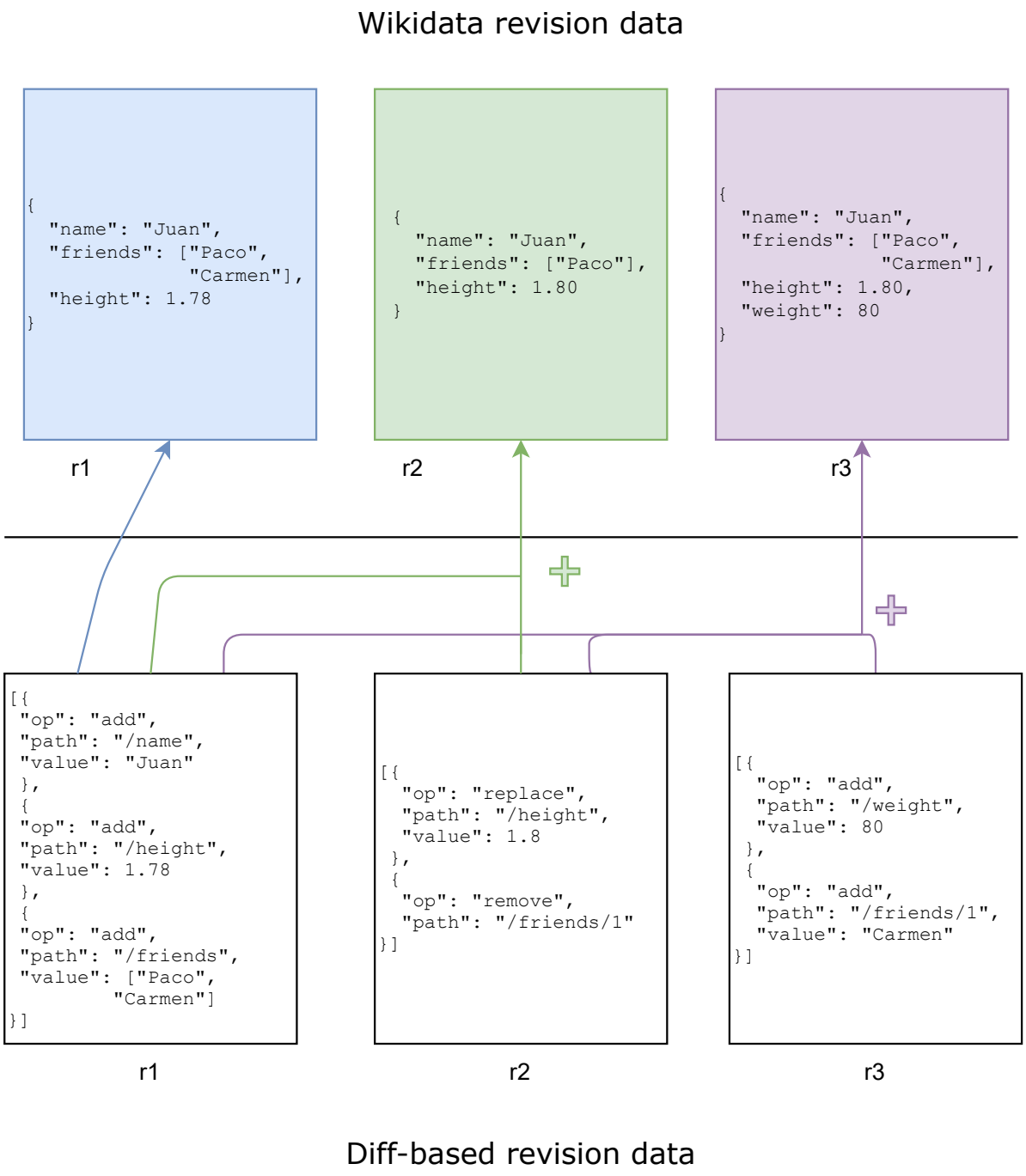


Figura 5.1. Ejemplo de descomposición de revisiones en JSON Patch. En la parte de arriba se muestra un ejemplo simplificado de entidad de Wikidata en cada revisión r. En la parte de abajo se muestra la descomposición de esta entidad en diffs JSON.

Cuadro 5.2. Estadísticas del dataset de ediciones indexado

Tamaño del dataset:	401GB
Número de entidades:	9.709.099
Número de revisiones:	443.116.607
Número de operaciones:	986.678.861

- **comment**: Comentarios de la revisión, si existen.
- **entity_diff**: Diff de la revisión, expresado en el formato JSON Patch explicado previamente.

5.3. Indexado de datos

Los datos con los diffs de cada entidad obtenidos fueron posteriormente indexados en una base de datos con el fin de optimizar las operaciones de consulta sobre éstos. Se tuvieron en cuenta distintas alternativas para almacenar el dataset, optando al final por utilizar una base de datos MongoDB. Las razones por las que se utilizó MongoDB fueron las siguientes:

- Los datos de cada entidad de Wikidata ya se encuentran en JSON, encajando en bases de datos basadas en documentos que utilizan JSON como MongoDB. Esto nos permite aprovechar el sistema basado en documentos JSON para no tener que crear un nuevo esquema ni transformar los ficheros de entrada, así como poder realizar consultas directamente sobre estos archivos JSON utilizando las capabilidades de MongoDB.
- El propio modelo de datos de Wikidata no sigue un esquema fijo, sino que es flexible en función de la entidad. Esto dificultaría mucho el convertir estos datos en un conjunto de tablas estructuradas, propias de un sistema de datos relacional. Con un sistema de datos basado en documentos podemos trabajar con esta flexibilidad y falta de estructura propia de los datos de Wikidata con los que estamos trabajando.
- Dentro de los sistemas de gestión de bases de datos basados en documentos disponibles, MongoDB cuenta con la mayor comunidad y drivers disponibles para diversos lenguajes de programación. Esto facilita la consulta de problemas y dudas a lo largo del proyecto.

En la tabla 5.2 se muestran las estadísticas principales del dataset indexado en MongoDB. Debido al gran volumen de datos con el que se trabaja, y las restricciones a nivel computacional del proyecto, se estableció una arquitectura de base de datos que optimizara la ejecución de consultas a realizar de cara a este proyecto. En primer lugar, se dividió la base de datos en dos colecciones: una de **entidades**, donde se almacena el contenido JSON completo de cada entidad tras su última revisión; y otra de **revisiones**, donde se almacenan los diffs de cada entidad tras cada revisión y sus metadatos correspondientes. Aunque el contenido final de cada entidad se puede obtener aplicando los diffs de sus revisiones, como hemos explicado previamente, el tener esta información redundada en una colección propia nos permitió poder estudiar tanto los datos de revisiones como los de las entidades en su estado final de forma más eficiente.

Por otro lado, se añadió un campo a cada revisión que incluye las clases a las que pertenece la entidad afectada. Esto nos permitió realizar consultas más eficientes que filtraran los datos por la clase afectada, a coste de aumentar el tamaño del dataset. Se establecieron índices en los campos de **id de la entidad** y en el de **clases de la entidad** ya que la mayoría de consultas empleadas filtraban las revisiones por esta información.

5.4. Extracción de información del dataset

Una vez indexados los datos de ediciones en MongoDB, el siguiente paso fue realizar una serie de análisis sobre este dataset para explorar este tipo de información dinámica.

Merece la pena explicar brevemente cómo se ha realizado el cálculo de las propiedades, entidades, y clases consideradas más 'conflictivas' de Wikidata. Este índice de conflictividad se obtuvo a partir del número medio de guerras de edición que experimenta el objeto entre el número de revisiones totales que tiene. En el contexto de nuestro proyecto consideramos que una guerra de edición se produce cuando una propiedad adquiere un valor x, alguien elimina o modifica su valor a otro valor distinto y, y posteriormente otro usuario vuelve a establecer el valor original x.

Otra información extraída incluye el desglose de operaciones realizadas en cada clase, y la obtención de propiedades más modificadas por decil de cada clase, entre otras cosas. Todos estos resultados obtenidos se muestran y explican en detalle en el capítulo 6.

5.5. Creación de modelos predictivos

5.5.1. Definición de tarea

Con el dataset de historial de ediciones de Wikidata obtenido y analizado, el siguiente paso fue proponer una tarea a modo de ejemplo sobre la que analizar el posible impacto del uso de esta información para el refinamiento de grafos de conocimiento. De cara al proyecto se analiza el uso del historial de ediciones para ayudar en tareas de predicción del tipo de una entidad. De forma concreta, en el caso específico de Wikidata esto equivale a predecir el valor de la propiedad P31 de Wikidata para un sujeto dado[7].

5.5.2. Técnicas propuestas

Para analizar el impacto de utilizar el historial de ediciones sobre las tareas definidas, partiremos de una de las técnicas más utilizadas actualmente para este tipo de tareas: el uso de representaciones vectoriales de cada uno de los nodos del grafo de conocimiento (*knowledge graph embeddings*). Para más información sobre esta técnica, se recomienda consultar la sección 2.2 Modelos basados en embeddings. Mientras que estos modelos suelen utilizar grafos 'estáticos' (es decir, sin incorporar información de las ediciones que experimentó el grafo hasta llegar a su versión final), nuestro objetivo será incorporar información de ediciones en estos modelos para intentar obtener una mejoría en su rendimiento. Para ello, se proponen las siguientes dos técnicas en este proyecto.

Generación de corrupciones a partir del historial de ediciones

En el proceso de generación de vectores de estos modelos se lleva a cabo una tarea denominada *'negative sampling'* en la que se generan tripletas falsas –*corrupciones*– que el modelo debe evitar sugerir. Esta generación se lleva a cabo de manera aleatoria, bajo la premisa de que el espacio total de tripletas posible ($|T|$) es mucho mayor que el de tripletas correctas, y por lo tanto las probabilidades de introducir *falsos negativos* (tripletas que consideramos como falsas para el modelo, pero en realidad no lo son) son bajas.

Sin embargo, en el historial de ediciones de cada entidad tenemos información potencialmente útil para poder determinar las corrupciones a generar: tripletas que usuarios de la comunidad han considerado como no válidas en algún momento del ciclo de vida de la entidad. Por ello, proponemos una serie de métodos de generación de corrupciones a partir del historial

[7]La propiedad P31 ('instancia de') en Wikidata se corresponde con *rdf:type* en el vocabulario RDF, y se utiliza a nivel ontológico para determinar la clase a la que pertenece una entidad.

de ediciones. Podemos sacar una analogía de esta idea con el proceso de entrenar un modelo que automáticamente realice *spell checking*. Una opción posible sería añadir, modificar, o eliminar caracteres de cada palabra del conjunto de entrenamiento para generar artificialmente errores (método de muestreo negativo básico). Sin embargo, una posible forma de mejorar este enfoque podría ser el incluir en este dataset de entrenamiento errores reales realizados por humanos al escribir esas palabras en sus dispositivos (método de muestreo negativo basado en el historial de ediciones).

Antes de explicar los métodos, es necesario definir una serie de elementos utilizados en el proceso de generación de corrupciones. El conjunto de tripletas eliminadas en el historial de ediciones será denominado Op^-, mientras que al conjunto de tripletas añadidas lo denominaremos Op^+. Cada uno de estos conjuntos contiene tripletas $T = (s, p, o)$, donde s es el sujeto de la tripleta, p el predicado, y o el objeto. El número de corrupciones que el modelo debe generar para cada tripleta será denominado n.

Tras esta preparación, pasamos a describir los métodos propuestos:

- **Muestreo negativo basado en el historial de ediciones**. En primer lugar, seleccionamos el conjunto de tripletas negativas para cada tripleta que sea necesario corromper. Este conjunto de tripletas negativas se obtiene a partir de todas las eliminaciones del grafo de conocimiento que compartan el mismo sujeto y predicado que la tripleta positiva que hay que corromper. Por ejemplo, si queremos generar corrupciones para la tripleta '`DouglasAdams ocupación escritor`', buscaríamos las tripletas eliminadas del historial de ediciones que tuvieran a '`DouglasAdams`' como sujeto y '`ocupación`' como predicado[8]. Posteriormente, se baraja el conjunto de tripletas negativas y se seleccionan los n primeros resultados como corrupciones generadas. En caso de que el conjunto de tripletas negativas tenga menos de n elementos, las corrupciones restantes se generan de manera aleatoria, siguiendo el proceso tradicional. Una variante de esta técnica consiste en generar el conjunto de tripletas negativas sin incluir aquellas tripletas que hayan sido objeto de guerras de edición. A lo largo de los experimentos se denominarán **'edit hist'** y **'edit hist (no edit wars)'** a los modelos que utilizan este tipo de generación de corrupciones incluyendo guerras de ediciones y omitiéndolas, respectivamente.

- **Muestreo negativo inverso basado en el historial de ediciones**. Con este método se parte de una hipótesis contraria a la anterior: las tripletas eliminadas en el historial de ediciones representan un conflicto de opinión sobre el valor de una propiedad, y no que el valor sea incorrecto. Por lo tanto, habría que evitar que esas tripletas eliminadas se añadan a las corrupciones del modelo, ya que son tripletas que pueden considerarse correctas o, al menos, parcialmente relacionadas con la entidad en cuestión. Por lo tanto, aunque partimos del mismo conjunto de tripletas negativas que hemos enseñado previamente, en este caso lo que buscamos es evitar que se genere una corrupción que pertenezca a este conjunto. Las corrupciones se generan de manera aleatoria, siguiendo el enfoque tradicional, pero en caso de que la tripleta generada pertenezca al conjunto de tripletas negativas se vuelve a lanzar el proceso de generación de corrupción hasta que la tripleta no pertenezca al mismo.

En los algoritmos 1 y 2 se muestra el proceso de obtención de corrupciones y ambos métodos de muestreo negativo propuestos previamente.

[8] En los ejemplos se muestra la corrupción de objetos en cada tripleta. El caso de corrupción de sujetos sería análogo, pero en este caso seleccionando como tripletas negativas aquellas que compartan predicado y objeto con la tripleta a corromper.

Algoritmo 1: Obtención de corrupciones a partir del historial de ediciones

Input:

$t = (s, p, o)$ Tripleta que va a ser corrompida.
$Op^- = \{t_1, t_2 ... t_m\}$ Conjunto de tripletas eliminadas del grafo.
$omit\_edit\_wars \in \{True, False\}$ Si las guerras de edición deberían omitirse o no.

Variables:

$F \subset Op^-$ Conjunto de corrupciones filtradas.

Output:

$T_{neg} \subset F$ Conjunto de muestras negativas de t.

begin

$F \longleftarrow \emptyset$
for $t' = (s', p', o') \in Op^-$ **do**
  if $omit\_edit\_wars = False$ *or not* $HasEditWars(t')$ **then**
    $F \longleftarrow F \cup \{t'\}$
$T_{neg} \longleftarrow \{(s', p', o') \in F : s' = s, p' = p\}$
return T_{neg}

Transformación de tarea no supervisada a supervisada

Mientras que la generación de *knowledge graph embeddings* es una técnica no supervisada en la que no se necesita etiquetar ninguna de las tripletas del grafo, al disponer del historial de ediciones podemos incorporar una nueva capa por encima de estos embeddings para convertirlo en una tarea supervisada. Es posible utilizar los vectores de cada tripleta como entrada a un modelo predictivo supervisado (p.ej., un clasificador basado en los k vecinos más cercanos), que entrenamos utilizando como etiquetas información del historial de ediciones.

De forma más precisa, si tenemos un conjunto de tripletas eliminadas Op^- y otro conjunto de tripletas añadidas Op^+ en el historial de ediciones, para cada tripleta $t \in Op^-$ podemos asignarle una etiqueta $y = 0$ mientras que a cada tripleta $t \in Op^+$ se le asignaría una etiqueta $y = 1$. De esta forma se puede entrenar un modelo predictivo a partir de la representación vectorial de cada tripleta y su etiqueta correspondiente obtenida del historial de ediciones. En nuestro caso, la entrada del modelo predictivo será la concatenación de la representación vectorial del sujeto, predicado, y objeto de la tripleta.

5.5.3. Obtención del dataset

Los modelos que acabamos de describir trabajan sobre un dataset en forma de grafo, mientras que el dataset indexado en MongoDB se encuentra en formato JSON. Para poder ejecutar estos modelos fue necesario llevar a cabo una conversión de estos datos a RDF, que se mostrará a continuación.

El primer paso en este proceso fue transformar el JSON 'estático' de cada entidad a RDF. Para ello, se transformaron tan solo los valores simples de cada *declaración* de las entidades (consultar sección de 2.1.1 para más información). Dicho de otra forma, en el proceso de conversión de JSON a RDF tenemos en cuenta tan solo las ediciones a propiedades y objetos de una entidad, sin tener en cuenta otros valores complejos como referencias o calificadores. Este es el proceso normalmente llevado a cabo para generar datasets de grafos de conocimiento en Wikidata, ya que todavía se está trabajando en que los modelos de *knowledge graph embeddings* puedan obtener buenos resultados si se usan tripletas reificadas [44].

Debido a que el dataset JSON indexado en MongoDB contaba con millones de tripletas, de cara al estudio realizado se seleccionó un subconjunto de datos para convertir a RDF. Este

Algoritmo 2: Métodos de muestreo negativo propuestos

Input:

$n \in \mathbb{N}$: Número de muestras negativas a generar por cada tripleta.
Op^- Conjunto de tripletas eliminadas del grafo.
$T = \{t_1, t_2, ..., t_m\}$ Conjunto de tripletas a corromper.
$omit\_edit\_wars \in \{True, False\}$ Si las guerras de edición deberían omitirse o no.

Variables:

T_{neg} Conjunto de muestras negativas de t.

Output:

$C_{out} \subset (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{B})$ Corrupciones generadas.

Function *InverseNegativeSampling(T, Op^-, omit_edit_wars, n)*:

$C_{out} \longleftarrow \emptyset$
for $t = (s, p, o) \in T$ **do**
  $T_{neg} \longleftarrow FetchCorruptions(Op^-, t, omit\_edit\_wars)$
  for $i = 1 : n$ **do**
    do
      $t' \longleftarrow RandomCorruption(t)$
    while $t' \notin T_{neg}$
    $C_{out} \longleftarrow C_{out} \cup t'$
return C_{out}

Function *EditHistoryNegativeSampling(T, Op^-, omit_edit_wars, n)*:

$C_{out} \longleftarrow \emptyset$
for $t = (s, p, o) \in T$ **do**
  $T_{neg} \longleftarrow FetchCorruptions(Op^-, t, omit\_edit\_wars)$
  while $|T_{neg}| < n$ **do**
    $T_{neg} \longleftarrow T_{neg} \cup RandomCorruption(t)$
  $T_{neg} \longleftarrow Shuffle(T_{neg})$
  $C_{out} \longleftarrow C_{out} \cup \{t_1, t_2...t_n : t_i \in T_{neg}\}$
return C_{out}

subconjunto cuenta con 300 entidades de cada una de las clases indexadas en MongoDB, dando un total de 390.000 tripletas en el grafo final. Consideramos que este dataset tiene un tamaño apropiado para validar nuestras contribuciones, teniendo en cuenta también las restricciones de recursos computacionales y tiempo del proyecto.

Para llevar a cabo las tareas definidas hemos creado dos representaciones distintas de cada uno de los grafos RDF de entidades. La primera es una representación 'estática', donde las tripletas representan el contenido en RDF de cada entidad. Esta representación es la utilizada en tareas refinamiento de grafos de conocimiento, y no incluye ningún tipo de información sobre las ediciones de cada entidad. La segunda es una representación 'dinámica', que contiene una serialización RDF de las operaciones llevadas a cabo a una entidad hasta llegar a su estado final.

Por último, se dividieron ambos datasets (estático y dinámico) en conjuntos de entrenamiento, validación, y prueba. Mientras que una división 'tradicional' implica separar un conjunto de entidades para cada uno de estos datasets, al disponer del historial de revisiones es posible realizar una división más apropiada a un caso de uso real en Wikidata. Para esta división se mantienen todas las entidades en cada uno de los conjuntos, pero en el dataset de entrenamiento se introduce el estado de cada entidad hasta un momento de tiempo t, en el de

Cuadro 5.3. Hiperparámetros optimizados para cada modelo no supervisado

Modelo	Embedding dim.	Epochs	Batch size	Learn. rate	Negatives
RotatE	768	13	64	0.009	5
TransE	64	10	521	0.024	7
MuRE	150	21	256	0.088	28

validación su nuevo estado desde el momento de tiempo t_i hasta el t_{i+n}, y en el de prueba su nuevo estado desde el momento de tiempo t_{i+n} hasta la actualidad. En nuestro caso particular, en el conjunto de entrenamiento se incluyeron el 70 % de las revisiones de cada entidad (en orden cronológico), en el de validación un 15 % adicional (entre el 70 % y el 85 %), y en el de prueba el último 15 % de revisiones.

Nota: Sobre la conversión del dataset dinámico a RDF

En principio todos los métodos de uso del historial de ediciones que hemos presentado podrían haber sido implementados a partir del dataset JSON indexado en MongoDB. Sin embargo, se ha decidido obtener una representación RDF de este dataset para los experimentos por los siguientes motivos:

- Para permitir reutilizar ambos datasets ('estático' y 'dinámico') unificando el formato de serialización, sin tener de gestionar datos en RDF para algunas tareas y datos en JSON para otras.
- Ambos datasets contienen un subconjunto de las entidades disponibles en MongoDB, acelerando en gran medida la búsqueda de ediciones de las entidades presentes en los mismos.
- Se plantea explorar en un futuro la creación de modelos de knowledge graph embeddings que exploten directamente las tripletas que contienen información de revisiones del grafo. Con este dataset se pretende poder comparar sistemas que trabajen sobre un modelo de datos RDF estático y uno que incorpore información del historial de ediciones del grafo.

5.5.4. Entrenamiento de modelos

Se han seleccionado 3 modelos de *knowledge graph embeddings* para evaluar el impacto de los generadores de corrupciones : RotatE, TransE, y MuRE. Estos modelos fueron seleccionados a partir del benchmark realizado por Ali et al. utilizando el framework Pykeen [45], teniendo en cuenta tanto el rendimiento de los mismos dentro de datasets similares a Wikidata como sus tiempos de entrenamiento.

Se optimizaron los hiperparámetros para cada combinación de modelo y estrategia de generación de corrupciones utilizando el dataset de validación. Dado que el espacio de posibles combinaciones de parámetros era demasiado grande como para ser explorado al completo, se realizó una búsqueda de parámetros aleatoria durante un periodo de tiempo. Cada combinación de modelo y generador de corrupciones fue optimizada durante 12 horas, para un total de 240h (10 días). Los parámetros finales obtenidos para cada modelo tras este proceso de optimización se muestran en la tabla 5.3. En cuanto a los parámetros específicos de cada modelo, en el caso de TransE se obtuvo un valor del parámetro $scoring\_fct\_norm$ igual a 2, mientras que para el modelo MuRE el mejor valor del parámetro l_p fue 2.

Nota: Validación cruzada en nuestro dataset

La validación cruzada es una técnica bastante común en el ámbito del aprendizaje automático para entrenar modelos aprovechando al máximo el conjunto de datos disponible mientras que a la vez se controla un posible sobreajuste del modelo. En esta técnica el dataset de entrenamiento se divide en k subconjuntos, y se realizan k iteraciones de entrenamiento donde 1 de esos subconjuntos se selecciona como conjunto de prueba y el resto como conjuntos de entrenamiento[a].

Mientras que este tipo de técnica funcionaría en una división del dataset por instancias, en el caso particular en el que se divide el dataset por momentos de tiempo no es posible su utilización. A modo de ejemplo, en una iteración de entrenamiento se podría seleccionar como conjunto de prueba el subconjunto 1, mientras que el resto de subconjuntos formarían los datos de entrenamiento. Esto no tendría sentido a la hora de entrenar el modelo ya que estaríamos probando como el modelo predice las primeras ediciones de una entidad a partir de las ediciones sucesivas, lo cual es el objetivo opuesto a nuestro planteamiento (predicir ediciones futuras)[b]. Por lo tanto, en nuestra optimización de parámetros se emplea un conjunto de entrenamiento y otro de validación fijos.

[a] Para más información sobre la validación cruzada se recomienda consultar el siguiente enlace: https://bit.ly/3mAleoP

[b] Este caso sería aplicable a cualquier otra elección de conjunto de validación, salvo cuando este se corresponde con el último subconjunto.

Tras optimizar los hiperparámetros se entrenó cada uno de los modelos sobre el conjunto de datos de entrenamiento combinado con el de validación, para aprovechar al máximo los datos disponibles en el dataset. Por lo tanto, el conjunto de entrenamiento final del modelo constituye el 85 % inicial de revisiones de cada entidad, y la evaluación se lleva a cabo sobre el último 15 % de revisiones de las mismas.

5.5.5. Evaluación

Cada uno de los modelos presentados devuelve un ranking ordenado de posibles valores para cada combinación de sujeto-predicado introducida como entrada. Dentro de este ranking los primeros resultados se corresponden con valores más probables o 'recomendados' por el modelo, mientras que los últimos resultados serían los menos probables. Para medir el rendimiento de cada modelo se utilizarán las siguientes métricas:

- **Ranking medio (MR)**: Con esta métrica se mide cuál es el ranking medio del valor correcto dentro de los valores devueltos por los modelos predictivos.
- **Ranking medio recíproco (MRR)**: El ranking recíproco se mide como $\frac{1}{ranking}$. Con esta métrica se mide la media de rankings recíprocos obtenidos por el modelo.
- **Aciertos en k (hits@k)**: Con esta métrica se mide el porcentaje de veces que el modelo devuelve el valor correcto dentro de los primeros k valores ofrecidos como resultado. En nuestros experimentos mediremos los **hits@1**, **5**, y **10**.

Dentro del conjunto de prueba hemos incluido aquellas entidades que tuvieran una nueva clase añadida al conjunto de prueba (o dicho de otra forma, un nuevo valor para la propiedad *instancia de*) en su último 15 % de revisiones. El conjunto de prueba final para esta tarea está compuesto por 4193 tripletas.

Posteriormente, se ejecutó cada modelo para obtener la lista ordenada de valores sugeridos para cada par sujeto-predicado existente en los datos de prueba y se midió el resultado para

cada una de las métricas mencionadas previamente. Hemos seguido el 'escenario filtrado' planteado por Bordes et al. en el proceso de evaluación, eliminando aquellas tripletas sugeridas por el modelo que ya hubieran aparecido en el conjunto de entrenamiento (*tripletas verdaderas*) [19].

Durante este proceso de evaluación se utilizaron tests de McNemar para medir si las diferencias entre los resultados ofrecidos por cada modelo eran estadísticamente significativas [46]. Para construir las tablas de contingencia se midió la precisión (hits@1) de cada par de modelos comparados, almacenando aquellos casos en los que ambos modelos acertaban correctamente el valor de la propiedad, los casos en los que ambos modelos fallaban, y los casos en los que uno de los modelos acertaba pero el otro fallaba. Por lo tanto, la hipótesis nula planteada en cada comparación es si ambos muestran una proporción de errores similares en el conjunto de prueba. Dado que nos interesan las diferencias existentes entre cada generador de corrupciones, sólo se realizaron comparaciones entre los generadores de corrupciones dentro de cada modelo probado. Se rechazará la hipótesis nula cuando se obtenga un $pvalue \leq 0,01$.

Para reportar el impacto de los generadores de corrupciones propuestos se sigue el enfoque de reporte de evaluaciones propuesto por Sparck-Jones [47]: las diferencias de rendimiento se consideran *no relevantes* si son menores de un 5 %; como *notorias* si se encuentran entre el $5 - 10$ %; y como *materiales* si son mayores del 10 %.

Adicionalmente, también se han realizado mediciones del tiempo de entrenamiento de cada combinación de modelo y generador de corrupciones. Para ello se ha seleccionado un conjunto de datos con 50.000 tripletas y se ha entrenado cada modelo 20 veces, midiendo los tiempos de entrenamiento en cada ejecución. Posteriormente se calculó la media de tiempo y los intervalos de confianza de 95 % para cada combinación a partir de estos datos.

5.6. Reproducibilidad

Los scripts de obtención del historial de ediciones se han implementado y ejecutado con Rust 1.5.8, y el indexado se ha realizado sobre una base de datos MongoDB 4.4. Los experimentos de creación de modelos predictivos se han llevado a cabo con Python 3.8, utilizando como librerías PyKEEN 1.8.0 [48], PyTorch 1.11.0 [49], pyRDF2Vec 0.2.3 [50], RdfLib 6.1.1, y Scikit-learn 1.0.2 [51]. Se ha utilizado el número 42 como semilla aleatoria en la creación del dataset RDF y el entrenamiento de los modelos.

El análisis del historial de ediciones se ha llevado a cabo en una máquina con un procesador Intel(R) Xeon(R) Silver 4214 CPU 2.20GHz de 16 núcleos, con 32GB de RAM. El entrenamiento y evaluación de los modelos se realizó en una máquina con un procesador Intel Xeon (Cascade Lake) de 4 núcleos, con 16GB de RAM y una tarjeta gráfica Tesla T4 16GB.

Se ofrecen una serie de Notebooks de Jupyter anexos a la entrega donde todo el código de experimentación realizado puede ser reproducido. Adicionalmente, se ofrecen imágenes de Docker para instalar todas las dependencias necesarias para la realización de los experimentos, junto con las versiones apropiadas de las mismas. Por último, también se ha publicado el volumen de Docker que contiene la base de datos con el subconjunto del historial de ediciones de Wikidata para que se pueda reutilizar en otras investigaciones.

Conclusiones

En este capítulo hemos explicado los experimentos realizados durante el trabajo. Estos se dividen en tres grandes bloques: la obtención de datos de ediciones de Wikidata y su indexado en una base de datos; el análisis de estos datos de ediciones; y el entrenamiento de modelos de predicción de tipos que utilizan el historial de ediciones. En el siguiente capítulo procederemos a mostrar e interpretar los resultados obtenidos tras llevar a cabo estos experimentos.

Capítulo 6

Análisis de Resultados

En este capítulo se muestran e interpretan los resultados obtenidos en el proyecto. Estos resultados se dividen en dos grandes bloques: el análisis de datos de ediciones realizado sobre Wikidata y la evaluación realizada sobre los modelos predictivos propuestos en este trabajo.

6.1. Exploración del historial de ediciones

Empezaremos por enseñar los resultados de la exploración del historial de ediciones de Wikidata. Mientras que Wikidata suele explorarse desde un punto de vista 'estático' (es decir, teniendo en cuenta tan sólo el estado del grafo de conocimiento en un instante de tiempo t), tener información sobre sus cambios a lo largo del tiempo nos permite entender mejor el comportamiento de la comunidad en la construcción del grafo.

6.1.1. Ciclo de vida de una entidad

En la figura 6.1 se muestra un grafo de transiciones que va desde una operación de edición realizada a una entidad hasta la siguiente. El ancho de cada arista representa el número de transiciones de una operación a la otra, mientras que el tamaño de cada nodo representa el número total de veces que se ha realizado dicha operación. Las cuentas de transiciones y operaciones se han obtenido a partir de la acumulación de las cuentas individuales de cada entidad del dataset. Con el fin de reducir el ruido presente en el grafo, se han eliminado del grafo aquellas transiciones de cada estado con una cuenta menor al 10 % de la cuenta total de transiciones salientes del mismo.

Si observamos el grafo podemos ver cómo la operación central del mismo es el añadir un grupo de declaraciones a una entidad, seguido de añadir referencias a un enunciado. También existe un triángulo de operaciones comunes en la esquina superior derecha del grafo, que representa el añadido de valores simples a una entidad: una declaración simple, referencia, o calificador. Ambos comportamientos son esperados, ya que el añadir referencias y calificadores a un enunciado que se acaba de crear es una práctica común en Wikidata.

Por lo general, las modificaciones y eliminaciones de elementos no son tan comunes como las operaciones de añadido. Sin embargo, estas operaciones suelen estar conectadas entre ellas (por ejemplo, una modificación o eliminación suele estar seguida de otra modificación o eliminación). Mientras que el añadir una declaración no implica una operación posterior que añada un calificador o una referencia, el eliminar la declaración si que implica automáticamente una eliminación de sus calificadores y referencias asociados. Por lo tanto, algunas de las conexiones existentes entre una operación de eliminado y la siguiente son esperadas.

Por último, aunque el cambio de rango y de orden de enunciados, referencias, y calificadores está permitido dentro de Wikidata, son operaciones bastante poco frecuentes dentro de la comunidad.

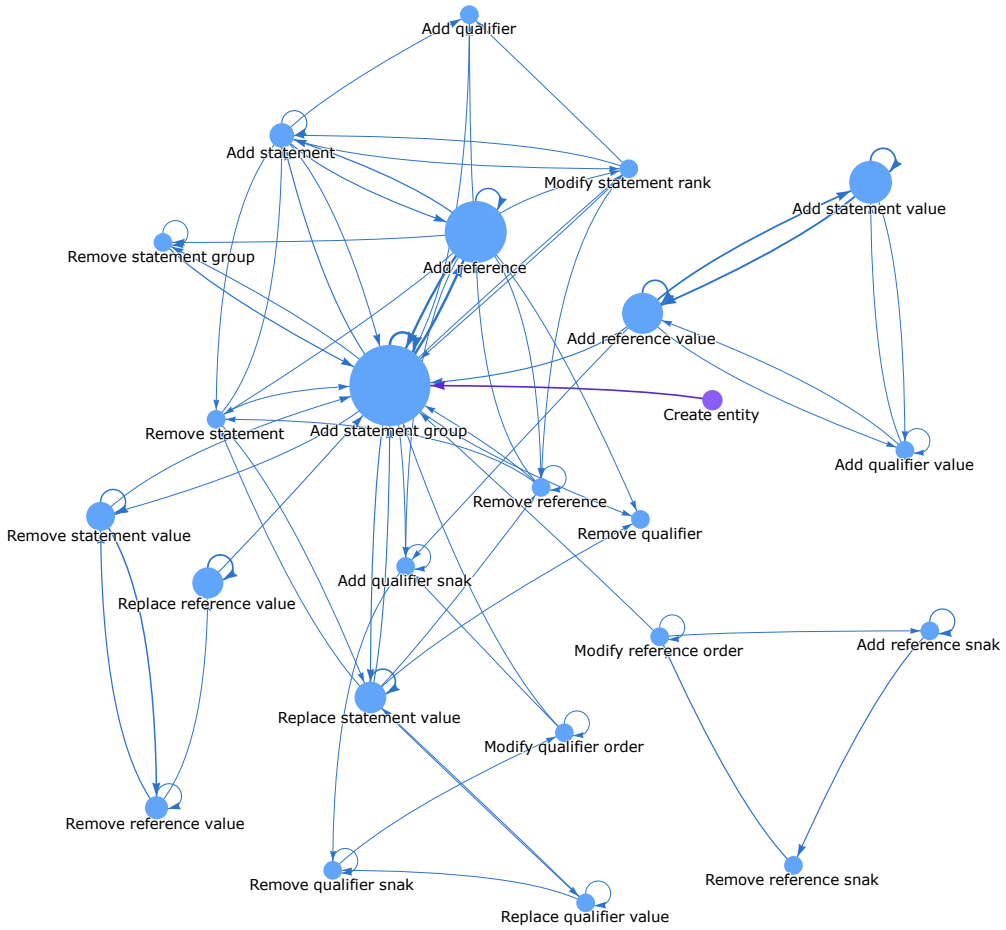


Figura 6.1. Transiciones más comunes de una operación de edición a la siguiente durante el ciclo de vida de una entidad. El círculo morado representa el comienzo del ciclo de vida de una entidad, y los círculos azules las diferentes operaciones aplicadas a la entidad.

6.1.2. Información sobre las ediciones

Operaciones realizadas sobre las clases más importantes

En la figura 6.2 se muestran las 10 clases con el mayor número de operaciones realizadas de media en sus instancias. Estas operaciones se muestran desglosadas por su tipo: añadidos, modificaciones, y eliminaciones. Se puede ver cómo las clases con más operaciones suelen tener un gran número de modificaciones de valores (superior al de añadidos y eliminaciones combinado), mientras que a medida que el número de operaciones disminuye las modificaciones tienen menor peso respecto a los añadidos. El ratio entre añadidos y eliminaciones puede ser superior o inferior dependiendo de la clase, lo que puede considerarse un indicador de la dificultad que tiene la comunidad en alcanzar un consenso sobre los valores de sus instancias.

Otro aspecto interesante a destacar de este gráfico es que las clases más importantes de Wikidata según su valor de ClassRank no tienen un gran número medio de operaciones por instancia. De hecho, las 2 clases más importantes (humano y taxón) ni siquiera aparecen dentro del top 10 de clases con más operaciones. Esto se puede deber a varios motivos. Una posible explicación sería que las instancias de estas clases no suelen contar con tanta información como las de otras clases, y por lo tanto suelen necesitar menos operaciones de media para ser representadas en Wikidata. Otra explicación posible sería que estas clases cuentan con un número de instancias muy grande y muchas de estas instancias pueden ser poco relevantes y

tener pocas operaciones, bajando la media de operaciones de la clase considerablemente con respecto a otras clases.

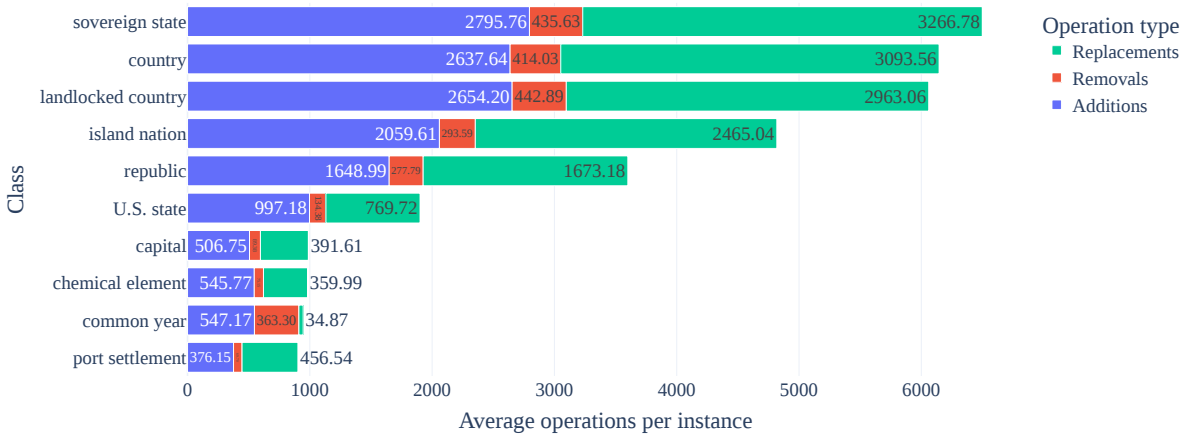


Figura 6.2. Top 10 clases con más operaciones realizadas de media sobre sus instancias.

Propiedades más eliminadas de una clase

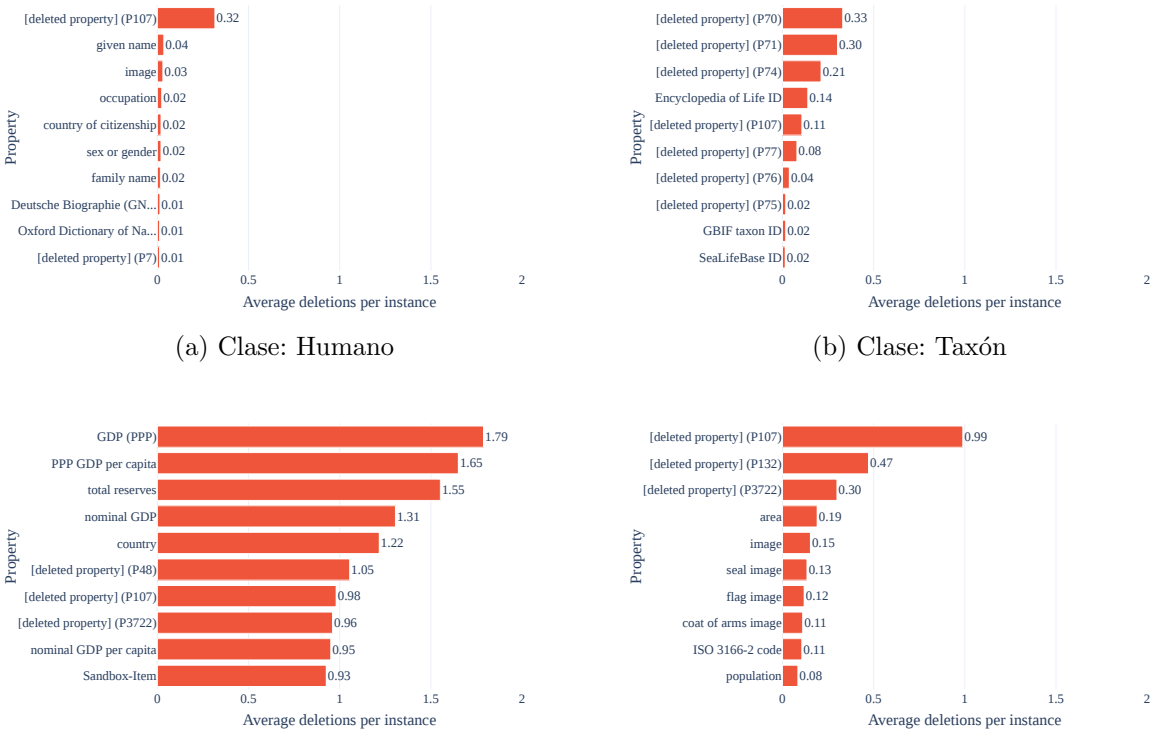
En la figura 6.3 se muestran las propiedades más eliminadas de 4 clases del dataset: *humano*, *taxón*, *estado soberano*, y *gran ciudad*. La etiqueta *[deleted property](PXX)* se refiere a propiedades que han sido eliminadas por completo de Wikidata, para las cuales no es posible obtener su etiqueta antes de la eliminación.

Podemos ver en la figura que hay algunas diferencias en los tipos de eliminaciones en función de cada clase. Para algunas clases, como *taxón* y *gran ciudad*, las propiedades más eliminadas son propiedades que se eliminaron de Wikidata. Estas propiedades no representan el nivel de consenso de sus valores dentro de la clase, ya que fueron eliminadas por completo de Wikidata desde un punto de vista ontológico. Las propiedades restantes tienen un ratio de eliminación más bajo, indicando que las propiedades usadas en estas clases son por lo general bastante estables y no suelen sufrir eliminaciones.

Otras clases, como *estado soberano*, tienen un alto número de eliminaciones en muchas de sus propiedades –las cuales siguen existiendo en Wikidata. Esto quiere decir que, en general, las instancias de esa clase tienen un conjunto de propiedades que o bien suelen generar conflicto dentro de la comunidad o que evolucionan de forma natural a lo largo del ciclo de vida de la entidad.

Por último, la clase *humano* tan sólo tiene una propiedad con más de $0,3$ eliminaciones por instancia, que ha sido eliminada de Wikidata. El resto de propiedades tienen un ratio de eliminación bastante bajo. Esto sigue el mismo patrón que hemos visto cuando analizamos el número de operaciones por clase, en el que aunque la clase *humano* sea la más importante en Wikidata –basándonos en ClassRank–, no tiene un alto número medio de operaciones por instancia.

Consideramos que un análisis en detalle de estos resultados sobre cada clase podría ser bastante beneficioso para poder entender qué propiedades son más controvertidas y cuáles no. Esta información podría ser utilizada por sistemas de recomendación que señalaran aquellas propiedades más controvertidas y que necesitan más atención, ayudando así a los editores.



(a) Clase: Humano

(b) Clase: Taxón

(c) Clase: Estado Soberano

(d) Clase: Gran Ciudad

Figura 6.3. Propiedades más eliminadas de cada clase.

Nota: Sobre las muestras presentadas en la exploración de datos

Aunque en algunos casos se muestren sólo resultados de algunas clases o entidades del dataset, todos los análisis aquí presentados se han realizado sobre el conjunto completo de clases y entidades del dataset. Sin embargo, debido a la gran cantidad de clases nos es imposible cubrir todas en un proyecto de este alcance. Con todos los datos obtenidos, es posible en un futuro realizar otros tipos de análisis agregados sobre la información y explorar en más detalle la información de Wikidata desde este punto de vista dinámico.

Conflicto en Wikidata

Nuestro último conjunto de análisis del historial de ediciones de Wikidata va alrededor de la idea de conflicto dentro de Wikidata. Hemos explorado el historial de ediciones para detectar aquellos casos en los que se produjera una *guerra de edición* entre los usuarios al editar una entidad. En el contexto de nuestro proyecto consideramos que una guerra de edición se produce cuando una propiedad adquiere un valor x, alguien elimina o modifica su valor a otro valor distinto y, y posteriormente otro usuario vuelve a establecer el valor original x. Por lo tanto, una guerra de edición puede representar tanto un acto de vandalismo que fue posteriormente revertido como un conflicto entre la comunidad al decidir el valor que una entidad debería tener para una propiedad dada.

En la figura 6.4 se muestran las propiedades con el mayor número de guerras de edición en el dataset. La propiedad con el mayor número es '*sexo o género*', con una diferencia notable respecto al resto de propiedades. Establecer un tipo a las entidades mediante el uso de la propiedad '*instancia de*' también suscita bastante conflicto, siendo el siguiente caso con el mayor número de guerras de edición. Las propiedades restantes son particulares de dominios

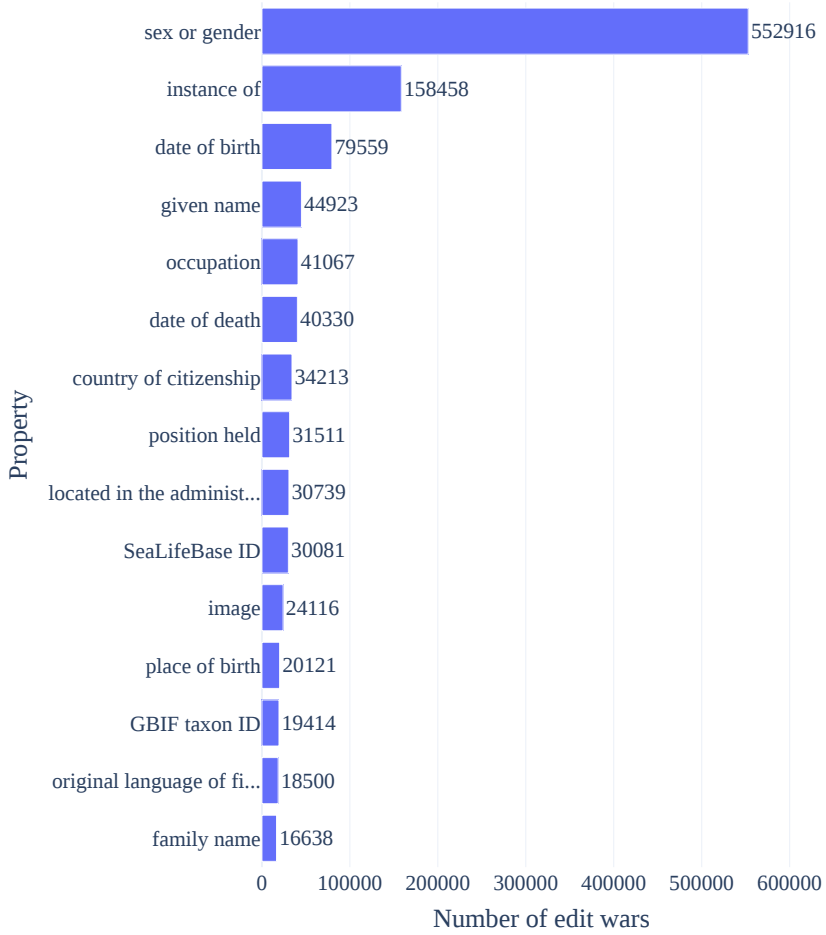


Figura 6.4. Top 15 propiedades con el mayor número de guerras de edición.

más concretos y tienen un número de guerras de edición considerablemente más pequeño comparadas con el top 2.

En la figura 6.5 se muestran las 15 clases con un mayor número de guerras de edición entre sus instancias. Podemos ver cómo la mayoría de guerras de edición ocurren en clases relacionadas con territorios, con tan sólo algunas pocas excepciones no pertenecientes a ese dominio (por ejemplo, elemento químico o lenguaje natural). De nuevo se vuelve a repetir el mismo patrón visto anteriormente, en el que la clase *estado soberano* cuenta con un gran número de eliminaciones y de modificaciones de valores en las propiedades de sus instancias.

Por último, en la figura 6.6 se muestran las entidades con más guerras de edición dentro de Wikidata. Podemos ver como entre las 15 entidades más conflictivas se encuentras 3 países, 2 ciudades y 6 personas. El caso de las ciudades y países ya fue tocado previamente, y sigue los patrones detectados hasta el momento. El caso de las personas también es esperado, ya que aunque pueda haber algunas personas que generen mucho conflicto en la comunidad la inmensa mayoría de personas tiene un contenido bastante estable, haciendo que en general no sea una clase muy conflictiva. Existen también 2 casos curiosos correspondientes a estadísticas de ciclismo femenino en 2018 y 2019 que tienen un alto número de guerras de edición y entraron en este top 15. Tras inspeccionar manualmente estas entidades se detectó que su contenido es generado casi en la totalidad por herramientas automatizadas, que entraron en un conflicto entre ellas en el que un bot añadía una seria de declaraciones y el otro las eliminaba periódicamente.

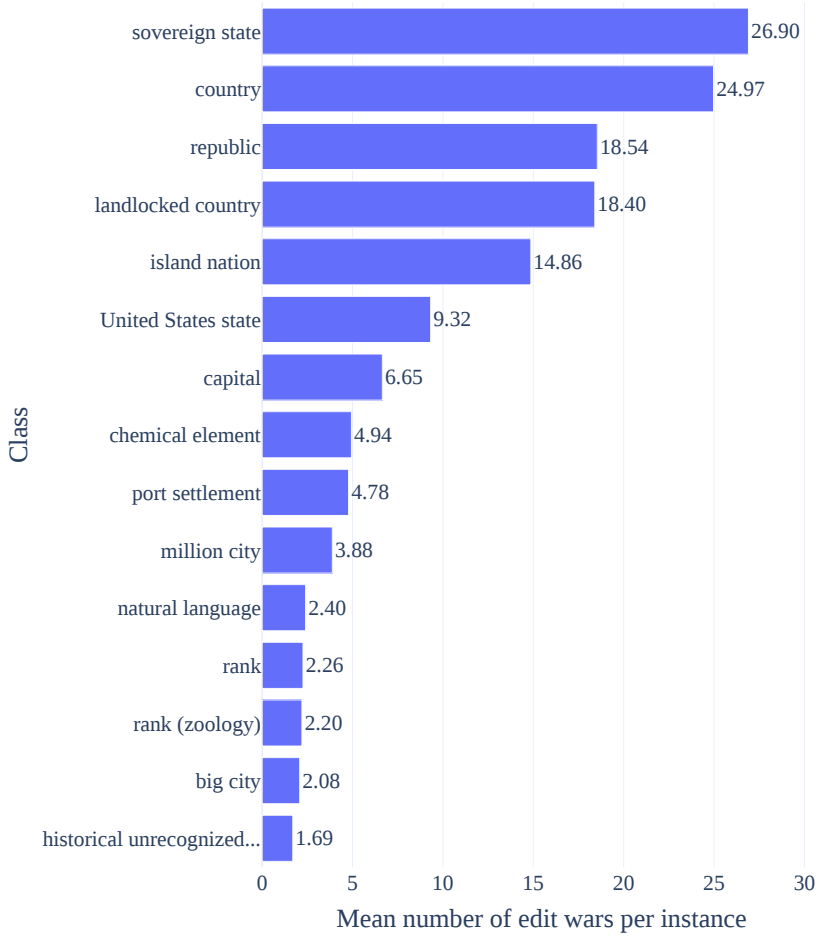


Figura 6.5. Top 15 clases con mayor número de guerras de edición.

6.2. Predicción de tipos

6.2.1. Evaluación de rendimiento

Generadores de corrupciones propuestos

Los resultados obtenidos para cada combinación de modelo no supervisado y método de generación de corrupciones se muestran en la tabla 6.1. Los resultados para la métrica de *hits@5* se muestran gráficamente en la figura 6.7.

Tras ejecutar el test de McNemar sobre cada par de generadores de corrupción se han obtenido las siguientes diferencias estadísticamente significativas para el modelo RotatE: *basic* frente a *edits* $[\chi^2(1) = 15,00,\ pvalue = 1 \cdot 10^{-4}]$; *basic* frente a *edits (no wars)* $[\chi^2(1) = 79,27,\ pvalue = 5,4 \cdot 10^{-19}]$; *basic* frente a *inverse* $[\chi^2(1) = 144,27,\ pvalue = 3,1 \cdot 10^{-33}]$; *edits* frente a *edits (no wars)* $[\chi^2(1) = 39,75,\ pvalue = 2,8 \cdot 10^{-10}]$; *edits* frente a *inverse* $[\chi^2(1) = 72,20,\ pvalue = 1,9 \cdot 10^{-17}]$.

En el caso del modelo TransE, obtenemos las siguientes diferencias estadísticamente significativas: *basic* frente a *edits* $[\chi^2(1) = 72,90,\ pvalue = 1,3 \cdot 10^{-17}]$; *basic* frente a *edits (no wars)* $[\chi^2(1) = 25,44,\ pvalue = 4,5 \cdot 10^{-7}]$; *basic* frente a *inverse* $[\chi^2(1) = 54,37,\ pvalue = 1,6 \cdot 10^{-13}]$; *edits* frente a *edits (no wars)* $[\chi^2(1) = 16,66,\ pvalue = 4,4 \cdot 10^{-5}]$;.

Por último, en el caso del modelo MuRE hay una diferencia estadísticamente significativa entre cada generador de corrupciones, con los siguientes resultados detallados: *basic* fren-

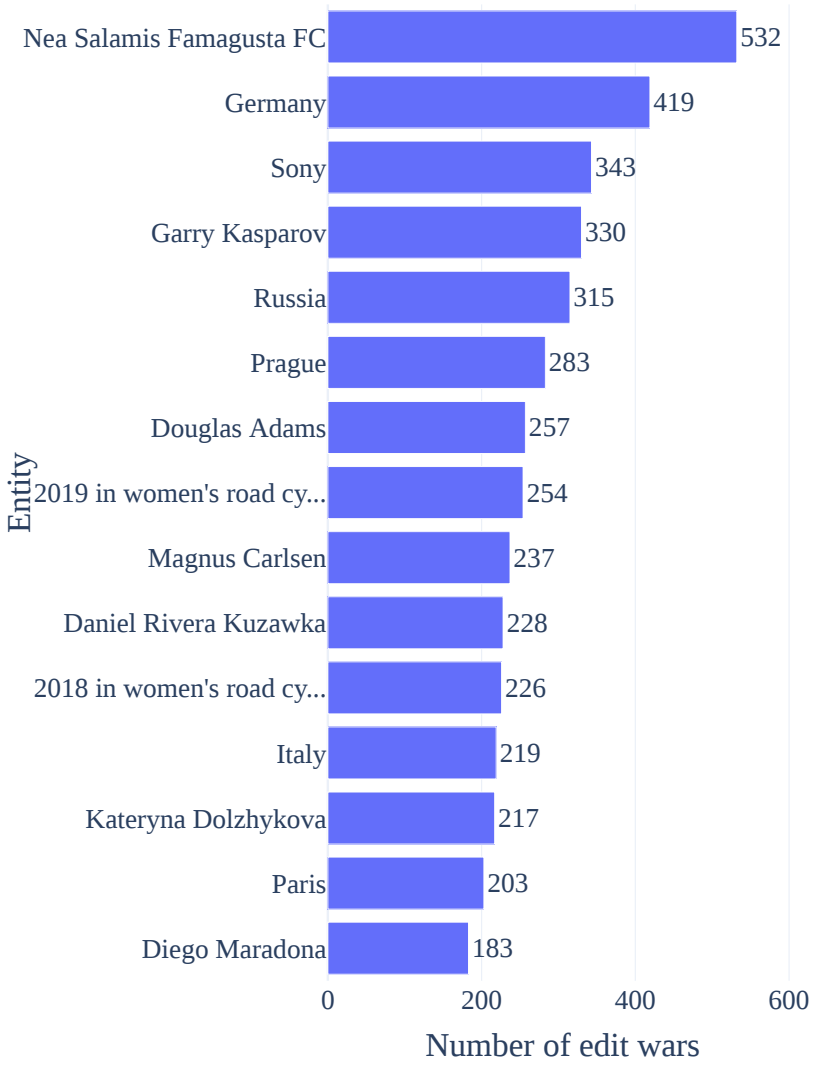


Figura 6.6. Top 15 entidades con mayor número de guerras de edición.

te a *edits* $[\chi^2(1) = 274,69,\ pvalue = 1 \cdot 10^{-61}]$; *basic* frente a *edits (no wars)* $[\chi^2(1) = 206,86,\ pvalue = 6,6 \cdot 10^{-47}]$; *basic* frente a *inverse* $[\chi^2(1) = 20,48,\ pvalue = 6,0 \cdot 10^{-6}]$; *edits* frente a *edits (no wars)* $[\chi^2(1) = 12,69,\ pvalue = 3,6 \cdot 10^{-4}]$; *edits* frente a *inverse* $[\chi^2(1) = 393,26,\ pvalue = 1,6 \cdot 10^{-87}]$; *edits (no wars)* frente a *inverse* $[\chi^2(1) = 307,65,\ pvalue = 7,1 \cdot 10^{-69}]$.

Las combinaciones restantes de generadores de corrupciones no muestran diferencias estadísticamente significativas ($p - value > 0,01$), con los siguientes resultados detallados: RotatE *edits (no wars)* y RotatE *inverse* $[\chi^2(1) = 6,01,\ pvalue = 0,014]$; TransE *edits* y TransE *inverse* $[\chi^2(1) = 2,58,\ pvalue = 0,10]$; TransE *edits (no wars)* y TransE *inverse* $[\chi^2(1) = 5,80,\ pvalue = 0,016]$.

Podemos observar a partir de los resultados que el uso de la técnica de generación de corrupciones '*inversa*' tiene un efecto positivo en todas las métricas para los modelos RotatE y MuRE, teniendo un impacto *material* y *notorio* respectivamente. Sin embargo, con el modelo TransE no ocurre lo mismo, ya que el generador de corrupciones básico mejora el rendimiento de todos los generadores propuestos en casi todas las métricas. La única excepción sería en la

Cuadro 6.1. Resultados de la evaluación de los modelos no supervisados. El generador de corrupciones básico que se utiliza actualmente en los modelos de knowledge graph embeddings se denomina '*basic*'; con '*edits*' y '*edits (no w.)*' nos referimos a los generadores de corrupciones propuestos basados en el historial de ediciones, que generan las muestras negativas a partir de las eliminaciones del grafo incluyendo u omitiendo guerras de edición respectivamente; por último, con '*inv.*' nos referimos al generador de corrupciones en el que las tripletas que fueron eliminadas del grafo de conocimiento no pueden ser generadas como corrupciones.

	RotatE				TransE				MuRE			
	basic	edits	edits (no w.)	inv.	basic	edits	edits (no w.)	inv.	basic	edits	edits (no w.)	inv.
MR	10819	21268	23577	**7197**	3204	**2385**	2756	2595	527	4650	3385	**447**
MRR	0.177	0.175	0.211	**0.260**	**0.091**	0.043	0.055	0.046	0.204	0.063	0.085	**0.237**
hits@1	0.086	0.109	0.146	**0.163**	**0.050**	0.015	0.028	0.020	0.115	0.018	0.031	**0.144**
hits@5	0.300	0.241	0.280	**0.382**	**0.121**	0.055	0.072	0.054	0.294	0.090	0.122	**0.330**
hits@10	0.382	0.308	0.339	**0.445**	**0.164**	0.091	0.097	0.083	0.385	0.148	0.210	**0.422**

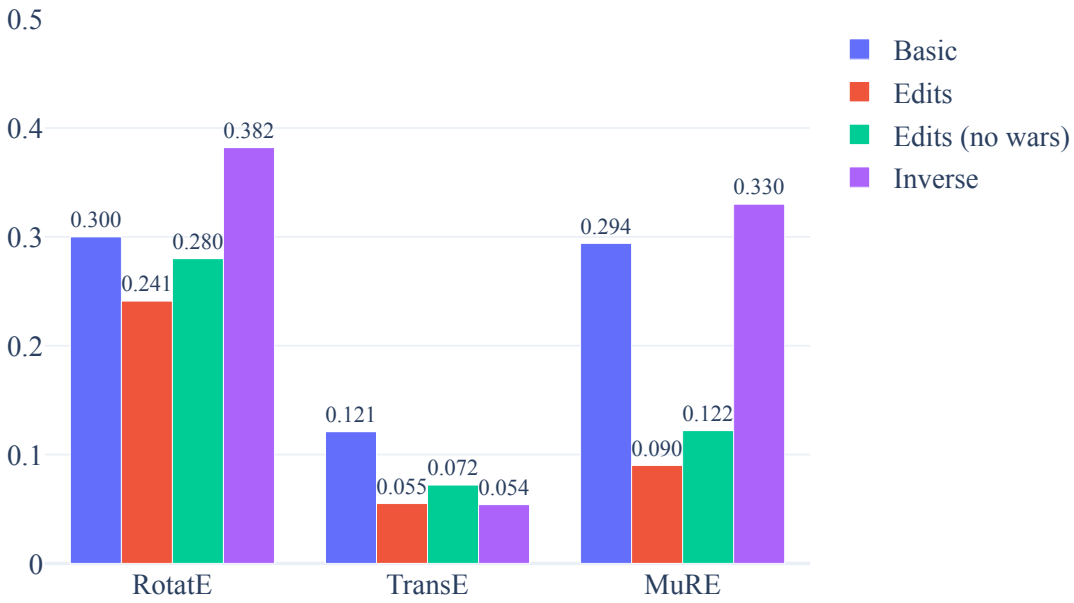


Figura 6.7. Resultados de hits@5 de cada modelo no supervisado.

métrica de ranking medio, donde el generador básico tiene un 24,68 % de ranking medio más alto en comparación con los enfoques propuestos. Cabe destacar que los resultados del modelo TransE son en general considerablemente peores que los obtenidos con los modelos MuRE y RotatE, independientemente del generador de corrupciones utilizado. Esto podría ser atribuido a no haber encontrado un conjunto de parámetros óptimo para el modelo, o a que directamente el modelo se comporta peor en este dataset frente a otras alternativas. En general, el generador de corrupciones basado en el historial de ediciones inverso que proponemos parece influenciar positivamente en los resultados, mejorando el rendimiento de 2 de los 3 modelos evaluados y obteniendo los 2 mejores resultados en términos de MRR y hits@k.

Por otro lado, en el caso de los generadores de corrupciones basados en el historial de ediciones ('*edit*' y '*edit (no wars)*') no se aprecia una mejora sobre el enfoque básico. Si consideramos los resultados del modelo RotatE, ambos generadores de corrupciones basados en el historial de ediciones todavía obtienen mejores resultados que el enfoque básico en la métrica de hits@1, pero las mejoras no son tan altas como en el caso de la propuesta '*inverse*'. Sin embargo, en el resto de modelos el rendimiento de estos enfoques es peor que el del generador de corrupciones básico en casi todas las métricas. Esto podría atribuirse a que, en alguunos casos, un valor que haya sido eliminado de una entidad puede ser en realidad verdadero –o, al menos, estar parcialmente relacionado con el dominio de la entidad que está siendo afectada.

Vamos a ilustrar esto último con un ejemplo real del dataset sobre el que estamos trabajando: la entidad que representa a *Buses MAN SE*[1] en Wikidata originalmente era una instancia de compañía[2], pero posteriormente este dato fue eliminado y modificado por la clase negocio[3]. Aunque existan detalles legales que diferencian a ambos términos, éstos se parecen mucho desde un punto de vista conceptual. En casos como este, generar la corrupción a partir de la tripleta eliminada (`Buses-MAN-SE instancia-de compañía`) podría añadir ruido al modelo. Esta también podría ser una causa del buen rendimiento del generador de corrupciones inverso, ya que evita estos casos.

También es importante entender que hay un componente aleatorio en cada uno de los métodos de generación de corrupciones a partir del historial de ediciones que hemos planteado. Dado que estamos obteniendo n tripletas negativas aleatorias para cada tripleta que tiene que ser corrompida, en algunos casos puede que estemos obteniendo tripletas que sean verdaderos negativos (por ejemplo, una edición que eliminó un acto de vandalismo) mientras que en otros casos podemos estar obteniendo tripletas negativas que tienen una alta similitud con la entidad de forma conceptual (como el ejemplo de Buses MAN SE mencionado previamente) o incluso la eliminación de una tripleta completamente válida. Esto puede explicar en parte la variabilidad en el rendimiento de los métodos de generación de corrupciones a partir del historial de ediciones, y abre camino a análisis adicionales que serán mencionados en la sección de trabajo futuro.

Otro aspecto interesante de los resultados es que omitir las guerras de edición es beneficioso para los resultados de los generadores de corrupciones que utilizan el historial de ediciones en todos los modelos. Eso es esperado desde un punto de vista intuitivo, ya que los valores que causan conflicto dentro de la comunidad no deberían ser añadidos –en general– como ejemplos de tripletas negativas al modelo. Estos valores suelen estar asociados con información que no está muy claro si la entidad debería tener o no, y por consiguiente no suelen estar relacionados con información incorrecta. Además, aunque estos valores hayan sido eliminados en el conjunto de entrenamiento, al haber pertenecido a guerras de edición es más probable que vuelvan a aparecer en el conjunto de tripletas añadidas del conjunto de prueba. Establecer estas tripletas como falsas en nuestros métodos de generación de corrupciones disminuiría su ranking sugerido en la fase de evaluación, empeorando por consiguiente el rendimiento del modelo.

[1] https://www.wikidata.org/wiki/Q708667
[2] https://www.wikidata.org/wiki/Q783794
[3] https://www.wikidata.org/wiki/Q4830453

Cuadro 6.2. Resultados de la evaluación entre cada mejor versión de los modelos no supervisados y el modelo supervisado.

	RotatE (inv.)	TransE (basic)	MuRE (inv.)	RDF2Vec + Random Forest
MR	7191	2385	**447**	15752
MRR	**0.260**	0.091	0.237	0.124
hits@1	**0.163**	0.050	0.144	0.062
hits@5	**0.382**	0.121	0.330	0.170
hits@10	**0.445**	0.164	0.422	0.249

Tarea supervisada

En la tabla 6.2 se muestran los resultados obtenidos por el enfoque supervisado frente a cada modelo no supervisado con su mejor generador de corrupciones obtenido previamente.

Los resultados de ejecutar el test de McNemar muestran una diferencia estadísticamente significativa entre los ratios de error de cada modelo, obteniendo los siguientes resultados detallados: RotatE frente a TransE $[\chi^2(1) = 15,00,\ pvalue = 1 \cdot 10^{-4}]$; RotatE frente a MuRE $[\chi^2(1) = 79,27,\ pvalue = 5,4 \cdot 10^{-19}]$; RotatE frente a RDF2Vec + Random Forest $[\chi^2(1) = 15,00,\ pvalue = 1 \cdot 10^{-4}]$; TransE frente a MuRE $[\chi^2(1) = 79,27,\ pvalue = 5,4 \cdot 10^{-19}]$; TransE frente a RDF2Vec + Random Forest $[\chi^2(1) = 79,27,\ pvalue = 5,4 \cdot 10^{-19}]$; MuRE frente a RDF2Vec + Random Forest $[\chi^2(1) = 79,27,\ pvalue = 5,4 \cdot 10^{-19}]$.

Como podemos ver en los resultados, el enfoque supervisado (RDF2Vec + Random Forest) es el peor enfoque en términos de ranking medio (**MR**) con una diferencia considerable, teniendo un ranking medio 34 veces superior al del mejor modelo (MuRE). Este gran descenso en ranking medio con respecto a otras métricas es esperado, ya que el modelo supervisado convierte la tarea de predicción de tipos en un problema de clasificación binario. Es decir, el modelo intenta maximizar la probabilidad de acertar correctamente si una tripleta es verdadera o no: el ranking interno de tripletas verdaderas y falsas no le importa al modelo.

Con respecto a otras métricas, su rendimiento es mejor que TransE pero peor que los modelos RotatE y MuRE. Esto sigue el mismo patrón que fue mencionado cuando se analizó el rendimiento de cada generador de corrupciones. Utilizar eliminaciones del historial de ediciones como falsas tripletas para el clasificador puede añadir ruido al sistema en algunos casos, disminuyendo el rendimiento general del modelo.

6.2.2. Tiempos de entrenamiento

En la figura 6.8 se muestran los resultados obtenidos del tiempo de entrenamiento de cada modelo y generador de corrupciones. En ninguno de los casos existe un solapamiento entre los intervalos de confianza de cada generador de corrupciones en cada modelo.

Podemos observar como en todos los casos el generador de corrupciones básico es el que menos tiempo tarda de entrenar entre todos los generadores propuestos. El generador de corrupciones basado en el historial de ediciones que omite las guerras de ediciones es el más lento en todos los casos. De media, este generador de corrupciones es un 61,18 % más lento que el generador de corrupciones básico.

En algunos modelos las diferencias de tiempo entre el generador de corrupciones básico y nuestras propuestas son mayores, mientras que en otros estas diferencias son menores. Una posible explicación a esto es que cada modelo tiene un número de corrupciones a generar distinto, que fue establecido en el proceso de optimización de hiperparámetros. De mayor a menor diferencia de tiempo de entrenamiento tenemos el modelo RotatE (número de corrupciones = 5),

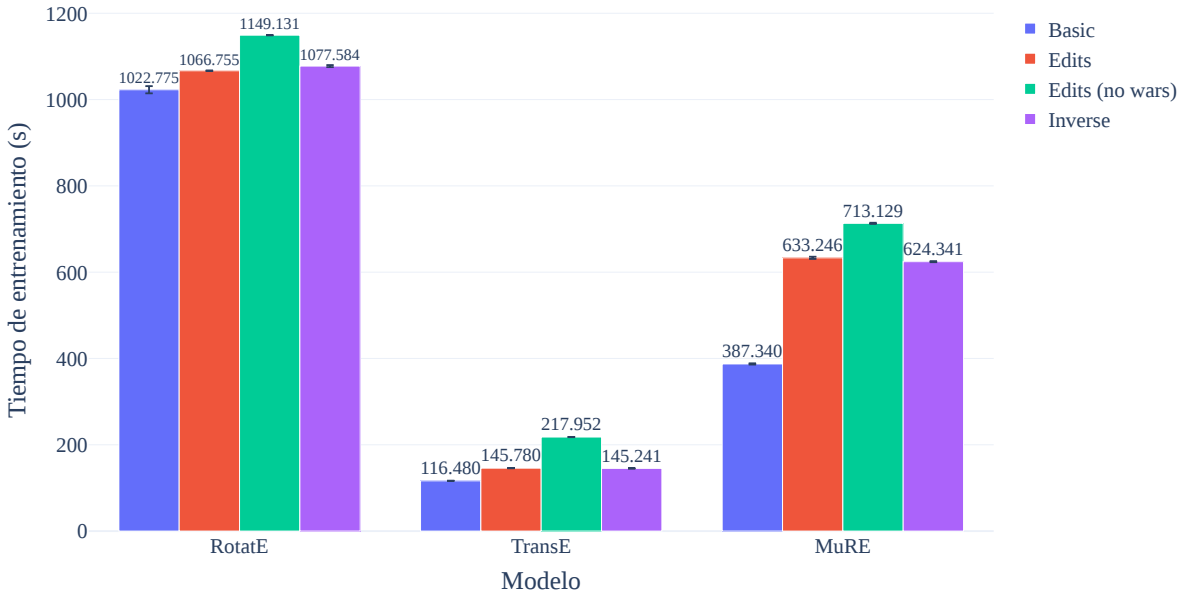


Figura 6.8. Tiempo de entrenamiento de cada modelo no supervisado.

seguido de TransE (número de corrupciones = 7), y finalmente MuRE (número de corrupciones = 28).

En general, las diferencias de tiempo obtenidas se deben a detalles de implementación de nuestras propuestas. Mientras que el generador de corrupciones básico genera las nuevas tripletas aleatorias de forma vectorizada, en nuestro caso esta generación no está optimizada. Una prueba de esto es que el generador de corrupciones que utiliza el historial de ediciones tarda más en entrenar al omitir las guerras de edición que cuando no las omite. Esto se debe a que al omitir las guerras de edición hay más casos en los que tenemos que generar tripletas aleatorias[4], que es donde más tiempo se pierde respecto a la implementación base. Por lo tanto, en un futuro sería interesante vectorizar el código de nuestras propuestas para obtener una mejor comparativa de estos enfoques respecto al generador de corrupciones base.

Aún así, teniendo en cuenta que el mejor modelo obtenido (RotatE con generador de corrupciones 'inverse') tan sólo tarda un 5,35 % más en entrenar respecto al modelo base consideramos que los resultados en cuanto a tiempo de entrenamiento son satisfactorios para la gran parte de casos de uso de estos sistemas. Es importante matizar que el tiempo de evaluación de los modelos es exactamente el mismo, ya que la única parte que se modifica en nuestras propuestas es el método de entrenamiento.

Conclusiones

En este capítulo hemos mostrado los resultados obtenidos en la investigación. Se ha analizado el ciclo de vida de las entidades dentro de Wikidata, así como los tipos de ediciones que se suelen realizar en función de la clase a la que pertenece la entidad. También se han interpretado los resultados de los métodos de predicción de tipos propuestos, detectando las ventajas e inconvenientes de utilizar el historial de ediciones. En muchos casos el hecho de que una tripleta haya sido eliminada no implica que esta sea incorrecta, lo que puede añadir ruido

[4]Haciendo vista atrás a los algoritmos definidos, esto se debe a que no tenemos suficientes tripletas negativas en el historial de ediciones y las tripletas restantes se generan de forma aleatoria.

a algunos modelos. En el siguiente capítulo trataremos las principales conclusiones del trabajo y nuevas áreas de trabajo futuro que han sido detectadas.

Capítulo 7

Conclusiones y Trabajo Futuro

En este trabajo hemos comenzado a explorar el el uso del historial de ediciones de un grafo de conocimiento para mejorar la calidad de los datos del mismo. Se ha construido un dataset con el historial de ediciones completo de las instancias de las 100 clases más importantes de Wikidata. Este dataset se ha analizado desde un punto de vista dinámico, descubriendo los comportamientos de edición presentes en Wikidata. Por último, se han propuesto una serie de métodos que incorporan el historial de ediciones en modelos de *knowledge graph embeddings* en la tarea de predicción de tipos, evaluando los aspectos positivos y negativos de incorporar esta información.

Las principales conclusiones del trabajo presentado son las siguientes:

- Existe una gran diferencia en los patrones de edición existentes en Wikidata en función de la clase a la que afectan. Algunas clases son susceptibles de mucho conflicto en su contenido entre la comunidad, mientras que otras permanecen bastante estables a lo largo del tiempo y no sufren apenas modificaciones.

- El uso de tripletas eliminadas del grafo como muestras negativas puede mejorar el rendimiento de un generador de corrupciones básico en algunos casos pero, por lo general, resulta en un peor rendimiento de los modelos. De media, con el generador de corrupciones básico se obtiene un 123 % de mejoría en el hits@1 en todos los modelos frente a un generador de corrupciones a partir de las tripletas eliminadas del grafo. Evitar las tripletas eliminadas que hayan sido sujeto de guerras de edición mejora el rendimiento de nuestro generador de corrupciones propuesto en todos los modelos evaluados. De media, eliminar estas tripletas mejora el hits@5 de nuestra propuesta en un 27,18 %.

- Por otra parte, el evitar la generación de corrupciones que pertenezcan al conjunto de tripletas eliminadas del grafo mejora el rendimiento de 2 de los 3 modelos, obteniendo los 2 mejores resultados entre todas las combinaciones de modelos y generadores de corrupciones evaluadas. Este enfoque mejora el hits@1 y hits@5 del enfoque básico en un 89,53 % y un 27,33 % respectivamente.

- El uso del historial de ediciones para entrenar un modelo supervisado de clasificación no mejora las técnicas existentes en la tarea de predicción de tipos. Se observan los mismos problemas que con el generador de corrupciones basado en el historial de ediciones, en que algunas de las tripletas eliminadas del grafo guardan relación semántica entre el sujeto y el objeto eliminado, y por lo tanto pueden añadir ruido al clasificador.

Trabajo futuro

Tras la realización de este proyecto se han detectado nuevas áreas de trabajo que sería interesante explorar en un futuro.

En primer lugar, se han detectado y señalado a lo largo de esta memoria varias formas de mejorar los enfoques de generación de corrupciones. Una de estas formas sería el añadir una puntuación a cada tripleta negativa que represente la 'idoneidad' de la tripleta para el modelo[1], y luego seleccionar las corrupciones a partir de esta puntuación. En cuánto al tiempo de entrenamiento de los modelos, se debería optimizar el código de los modelos vectorizando la generación de corrupciones y posteriormente comparar los tiempos con el enfoque básico para poder detectar las pérdidas de rendimiento de nuestras propuestas de forma más precisa. También sería interesante evaluar estas propuestas frente a sistemas de generación de corrupciones más complejos, como aquellos basados en redes generativas antagónicas (GANs), una vez que se retoquen los sistemas propuestos.

Tras el análisis de datos de ediciones realizado también surge la necesidad de poder diferenciar entre aquellas ediciones que se corresponden con vandalismo, un conflicto de opiniones entre la comunidad, o la evolución natural de una entidad. Tener un método que permitiera distinguir entre estos tipos de ediciones sería clave para tener un mejor conocimiento del comportamiento de la comunidad de Wikidata y para mejorar nuevos sistemas que hagan uso del historial de ediciones.

Por último, sería interesante explorar otras tareas de refinamiento de grafos de conocimiento donde se podría aprovechar la información del historial de ediciones. Por poner un ejemplo, se podría intentar la creación de modelos que aprovechen directamente la versión dinámica del dataset RDF proporcionado, trabajando directamente sobre el grafo que incluye las operaciones y revisiones realizadas. También existen sistemas de obtención automática de esquemas a partir de los contenidos del grafo [52] a los que sería interesante añadir información del historial de ediciones con el fin de poder generar esquemas o reglas menos genéricas.

[1] Sería necesario definir bien qué tripletas consideramos que son buenas para el modelo y cuáles son peores. De forma intuitiva, se deberían intentar evitar aquellas tripletas en las que, aunque hayan sido eliminadas, el sujeto y el objeto guardan una relación a nivel conceptual.

Apéndices

Apéndice A

Difusión de Resultados

A.1. Selección de revista

Se utilizaron dos herramientas principales de cara a la selección de revista JCR en la que enviar el artículo. Por un lado, la herramienta JournalFinder[1] que proporciona Elsevier para buscar artículos potencialmente relacionados con la temática del artículo. Por otro lado, el propio buscador de revistas JCR de Clarivate[2] para analizar las revistas que hayamos detectado con JournalFinder o con otros medios. A partir de estas herramientas se seleccionaron las siguientes 5 revistas JCR potenciales en las que enviar el artículo:

- International Journal on Semantic Web and Information Systems
- Journal of Web Semantics
- IEEE Transactions on Knowledge and Data Engineering
- Knowledge Based Systems
- Information systems

Posteriormente se analizaron las publicaciones de cada revista, buscando aquella en la que mejor podría encajar nuestro trabajo. Finalmente, se decidió enviar el artículo a la revista *Journal of Web Semantics*[3] (ISSN 1570-8268, Q3 en 2020 con un factor de impacto de 1,897) de Elsevier.

A.2. Artículo enviado

En las siguientes páginas se adjunta el artículo enviado a la revista.

[1] https://journalfinder.elsevier.com
[2] https://jcr.clarivate.com/jcr/home
[3] https://www.sciencedirect.com/journal/journal-of-web-semantics

Leveraging Wikidata's edit history in knowledge graph refinement tasks

Alejandro Gonzalez-Hevia[a,*], Daniel Gayo-Avello[a]

[a]*Department of Computer Science, University of Oviedo, Spain*

Abstract

Knowledge graphs have been adopted in many diverse fields for a variety of purposes. Most of those applications rely on valid and complete data to deliver their results, pressing the need to improve the quality of knowledge graphs. A number of solutions have been proposed to that end, ranging from rule-based approaches to the use of probabilistic methods, but there is an element that has not been considered yet: the edit history of the graph. In the case of collaborative knowledge graphs (e.g., Wikidata), those edits represent the process in which the community reaches some kind of fuzzy and distributed consensus over the information that best represents each entity, and can hold potentially interesting information to be used by knowledge graph refinement methods. In this paper, we explore the use of edit history information from Wikidata to improve the performance of type prediction methods. To do that, we have first built a JSON dataset containing the edit history of every instance from the 100 most important classes in Wikidata. This edit history information is then explored and analyzed, with a focus on its potential applicability in knowledge graph refinement tasks. Finally, we propose and evaluate two new methods to leverage this edit history information in knowledge graph embedding models for type prediction tasks. Our results show an improvement in one of the proposed methods against current approaches, showing the potential of using edit information in knowledge graph refinement tasks and opening new promising research lines within the field.

Keywords: Semantic Web, Wikidata, Edit History, Knowledge Graph Refinement, Type Prediction, Knowledge Graph Embeddings

1. Introduction

Different fields have incorporated the use of domain-specific knowledge graphs during recent years to solve their tasks. Some concrete examples of such domain-specific tasks include performing investment analysis [1], managing diseases and symptoms from medical records [2], or automatically generating test cases for software projects [3], among many others. Furthermore, the emergence of several open and general-purpose knowledge graphs, such as DBPedia [4] and Wikidata [5], has also attracted new communities closer to the Semantic Web by allowing them to exploit this structured information for many different applications. It goes without saying that most of those applications rely on the correctness and completeness of the data in the knowledge graph to deliver their results.

It is therefore crucial to ensure a high level of quality for those knowledge graphs. This has led to works that define quality metrics and dimensions to better analyze and understand data quality [6]. Those works reveal the existence of constraint violations and missing information in modern knowledge graphs, among other quality problems [7, 8].

Therefore, a number of different approaches have been proposed to improve the quality of knowledge graphs. Some of them follow a deductive approach, where a set of rules or constraints that each triple must follow are defined to enforce data quality [9, 10]. Other proposals follow an inductive approach, using predictive models or alternative probabilistic methods to try to fill incomplete information or fix errors in the knowledge graph [11].

However, in the specific case of collaborative knowledge graphs like Wikidata, there is an element that has not been fully explored yet: its edit history information. One of the main features of Wikidata, telling it apart from other open general-purpose knowledge graphs, is its collaborative approach: anyone can start from scratch editing entities in Wikidata. At the time of this writing, there have been 1,640,933,943 edits made to Wikidata. In those edits, the community has progressively built a consensus –fuzzy and somewhat distributed among the editors– over the information that best represents each entity within the knowledge graph, while also capturing the natural evolution of those entities across time.

In this paper we explore the possibilities of leveraging edit information to refine the contents of a knowledge graph, laying the foundations for future work in this area. Our main contributions are:

*Corresponding author

Email addresses: uo251513@uniovi.es (Alejandro Gonzalez-Hevia), dani@uniovi.es (Daniel Gayo-Avello)

URL: www.alejgh.com (Alejandro Gonzalez-Hevia), www.danigayo.info (Daniel Gayo-Avello)

1. The creation of a JSON dataset containing the complete edit history of every entity of the 100 most important classes in Wikidata, following Wikidata's data model (Section 3).
2. An analysis of the main editing patterns from contributors, edits made by class, and divisiveness in Wikidata based on the edit information (Section 4). This information is analyzed with a focus on its possible applications to knowledge graph refinement tasks.
3. The proposal of two approaches to leverage edit history data in type prediction tasks: the use of edits in the negative sampling process of knowledge graph embeddings models, and using the edit information as labeled data fed to a classifier (Section 5.1). We perform an evaluation of these approaches against a set of baselines and analyze the impact of using edit history information in both approaches.
4. An RDF dataset containing edit history information about Wikidata, following a custom data format where each operation and revision is serialized to the graph. This RDF dataset is also available without edit history information, and can serve as a baseline to measure the impact of using edit history data in knowledge graph refinement models (Section 5.2).

The rest of this paper is structured as follows. In the next section, we go over Wikidata's data model and provide a formal definition of a knowledge graph and its edits. These concepts are needed to better understand the successive aspects of our work. Section 3 goes over the process of acquiring edit information from Wikidata. This edit information is explored in section 4. In Section 5, we propose two new methods to leverage edit information to improve existing knowledge graph embedding models, and we evaluate their performance with respect to current approaches. Related work is reviewed in Section 6. And, finally, in Section 7 we present the conclusions of this work and future research directions.

2. Background

2.1. Wikidata data model

Entities are the basic building block of Wikidata's data model. There are two different types of entities: *items* and *properties*. Each entity is given a unique incremental numeric id, with items being prefixed by a 'Q' and properties by a 'P'.

A *statement* is composed of a property and a value assigned to that property, optionally having 1 to n qualifiers and 1 to n references. In the rest of this paper we will use the term *simple statement* to refer to statements that are just composed of a property and a value. A *statement group* is the set of statements that an item has of a given property. Each entity in Wikidata is composed of 0 to n statement groups.

Qualifiers are used to give further information about a given statement (e.g., the point in time when the statement holds true). Each qualifier is also composed of a property and a value assigned to that property. The combination of a property, value, and qualifier is called a *claim*. *References* are also property-value pairs, and they provide the source that validates a statement.

Aliases, descriptions, and labels constitute the *fingerprint* of an entity. These elements are mapped internally to `skos:altLabel`, `schema:description`, and `rdfs:label` URIs in the RDF representation of an entity. The combination of description and label of an entity in a given language must be unique. An entity can have multiple aliases but only a single description and label for a given language.

Snaks are the most basic information structure in Wikidata, and provide information about the value of a property. There are three types of snaks: *value*, *somevalue*, and *novalue*. Value snaks indicate that the property has a known value, which is then represented using Wikidata's available datatypes[1]. Somevalue snaks indicate that the property has a value but its value is unknown[2]. Finally, novalue snaks indicate that the property does not have a value.

Wikidata also introduces three ranks which can be assigned to each statement: `preferred`, `deprecated`, and `normal`. These ranks are generally used to decide which statements must be returned when querying Wikidata, and also to clean up its user interface when exploring an entity. Statements can also have an order within each statement group. Although the order of statements within each rank is not relevant, it can be changed by users.

All these elements are internally serialized in Wikidata to JSON and different RDF serialization formats. It must be noted that in this section we have covered all the elements that are mentioned in the rest of this paper, but the list is not exhaustive. Additional information about Wikidata's data model and its serialization is available online[3].

2.2. Revisions

Wikidata allows any user to edit entities. Therefore, each entity is composed of a *revision history* that holds all the changes made to the entity by Wikidata contributors. Any of the elements described in the previous section can be changed, and a single revision may hold changes to any number and combination of elements of an entity. Throughout this paper we will use the terms *edition* and *revision* interchangeably, following Wikidata's terminology.

[1] More information available at `https://www.wikidata.org/wiki/Special:ListDatatypes`

[2] Somevalue snaks are represented with blank nodes in the RDF serialization of the data model.

[3] `https://www.mediawiki.org/wiki/Wikibase/DataModel?tableofcontents=0`

A revision also contains additional metadata, including its timestamp, author of the revision, tags, and a description. Tags are usually used to indicate the device from which the revision was made (or the tool that made the edit, if it was an automated process) and also to indicate the cause of the edit[4] (e.g., to revert vandalism).

In the context of this paper we will use the term *operation* to refer to a single modification made to an entity's element in a revision. One revision may be composed of 1 to n operations. An operation may represent the addition or removal of a single element from the entity. For the sake of simplicity, we will also consider replacements as operations, which are a combination of an addition and removal operation to an entity.

Wikidata allows *restoring* and *undoing* revisions made to an entity. Restoring allows users to undo all the edits made to an entity up to the selected restoration state. Undoing is more versatile, since it undoes from 1 to n edits selected by the user, which do not need to be consecutive edits like in the restoration process. In none of those cases the revisions being undone are removed from the revision history of the entity. A new revision is made instead, which includes the necessary operations to undo the selected revisions.

Edits can be manually removed by Wikidata administrators under specific circumstances. These include revisions that contain private information, a violation of copyright, or personal attacks of a serious nature.

2.3. Formal definitions

We will now formally introduce the main elements used throughout this paper. Let $\mathcal{I}$, $\mathcal{L}$ and $\mathcal{B}$ be disjoint countably infinite sets of IRIs, literals and blank nodes respectively. A knowledge graph can be formally defined from a static point of view as a set of triples $(s, p, o) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{B})$.

From a dynamic point of view, a knowledge graph is built from a sequence of *operations* $Op = \{op_j : 1 \leq j \leq \infty\}$. Each operation op_j is composed of a triple $t = (s, p, o) : s \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{B})$. $Op^+ = \{t_1, t_2, ..., t_n\}$ represents the set of addition operations of the graph, while $Op^- = \{t'_1, t'_2, ..., t'_m\}$ represents the set of removal operations. The set of all operations is therefore defined as $Op = Op^+ \cup Op^-$. A knowledge graph is built out of n operations, with K_i representing the state of the graph after applying all operations up to operation i. Applying an addition operation $op^+_{i+1} = (s, p, o)$ to a graph K_i results in graph $K_{i+1} = K_i \cup (s, p, o)$. On the other hand, applying a removal operation $op^-_{i+1} = (s, p, o)$ to a graph K_i results in graph $K_{i+1} = K_i \setminus (s, p, o)$. The final state of a knowledge graph can be obtained by a successive application of all its operations.

[4] A list of the most common tags can be accessed at `https://www.wikidata.org/wiki/Special:Tags`

3. Extracting edit history data from Wikidata

We now present the approach followed to extract the edit history information from Wikidata to conduct our experiments.

3.1. Subset selection

Wikidata was composed of 97,795,169 entities at the time of this writing, with more than 1,640,933,943 revisions in total made by users[5]. Given that working with the entire Wikidata revision history could be too computationally expensive to validate our proposal, we extracted a subset to conduct our experiments.

This subset is composed of instances from the *most important* Wikidata classes. To that end, we have computed the ClassRank [12] score of every class in Wikidata, choosing the top 100 classes with the highest score. Then, we extracted the edit history information of every entity that is an instance of any of those classes. We preferred choosing the most important classes for our experiments over producing a random sample since –in general– entities belonging to central classes receive more attention from the community, and are therefore more promising for exploiting their edit history information.

To run ClassRank we defined the *P31* property (*instance of*) of Wikidata as a *class-pointer*[6]. The ClassRank score of a class is computed by aggregating the PageRank [13] scores of all its instances. However, since computing the PageRank score of every entity in Wikidata was too computationally expensive, we used a set of pre-computed PageRank scores. These scores were obtained using the Danker[7] tool, which periodically computes the PageRank score of every existing entity in Wikipedia [14]. The scores are then mapped from Wikipedia pages to their respective Wikidata entities, getting an approximation of their PageRank value[8].

The 20 most important classes based on their ClassRank score can be seen in table 1. These results were then filtered manually, since some of those entities could be considered Wikidata classes at the ontological level, but not at a conceptual one. To do so, we have removed those classes that contained the term "Wikimedia" in their labels, since they are used to organize Wikimedia content but do not represent classes at a conceptual level.

The final subset is composed of 89 classes and 9.3 million instances –around 10% of the total number of entities in Wikidata. Although this subset is composed of just a 10% of the entities in Wikidata, its size is around a 35% of the total size of Wikidata. This can be explained due to

[5] Source: `https://www.wikidata.org/wiki/Wikidata:Statistics`

[6] The *class-pointer* is used by ClassRank to fetch those entities from Wikidata that are classes.

[7] `https://github.com/athalhammer/danker`

[8] These dumps can be accessed at `https://danker.s3.amazonaws.com/index.html`

Table 1: Top 20 most important classes based on their ClassRank score

Name	ClassRank score	Number of instances
human	2,167,439	3,873,812
Wikimedia category	1,057,559	2,207,283
sovereign state	837,883	203
taxon	755,681	1,962,491
country	746,208	193
point in time with respect to recurrent timeframe	635,401	3,273
calendar year	499,551	666
big city	321,893	3,238
human settlement	321,637	512,417
Wikimedia disambiguation page	317,631	1,294,218
Wikimedia administration category	302,196	12,146
Wikimedia list article	287,605	301,370
language	284,433	8,894
modern language	257,324	6,875
city	247,022	8,650
academic discipline	204,426	1,603
time zone named for a UTC offset	170,254	72
metacategory in Wikimedia projects	167,871	2,545
republic	165,881	78
capital	161,384	388
taxonomic rank	160,822	67

the fact that the most important entities have, in general, more content introduced by the community with respect to lesser important classes.

3.2. Data extraction

Wikidata periodically releases public dumps of its contents[9]. We have selected the *pages-meta-history* dumps to extract the edit history of every entity from our subset, since these dumps are the only ones containing every revision of each entity and not just their final content. This dataset is composed of several XML files containing metadata of every revision made to each entity, and also a JSON blob with the complete content of the entity after each revision. Our final dataset is built from the *pages-meta-history* dumps from 2021-11-01.

Since working with the complete content of every entity after each revision leads to a lot of redundant entity information, we made some preprocessing steps to reduce the dataset size. Instead of storing the complete JSON content of each entity after every revision r, we computed the diff between the JSON content in the previous revision (r_{t-1}) and the current one (r_t). These diffs are stored in the JSON Patch format[10], therefore allowing the reconstruction of the entity contents after any revision. To obtain the complete JSON content of an entity at revision r_t we just need to apply the patches of every previous revision up to r_t. A simplified example of the decomposition of an entity in diffs is illustrated in figure 1.

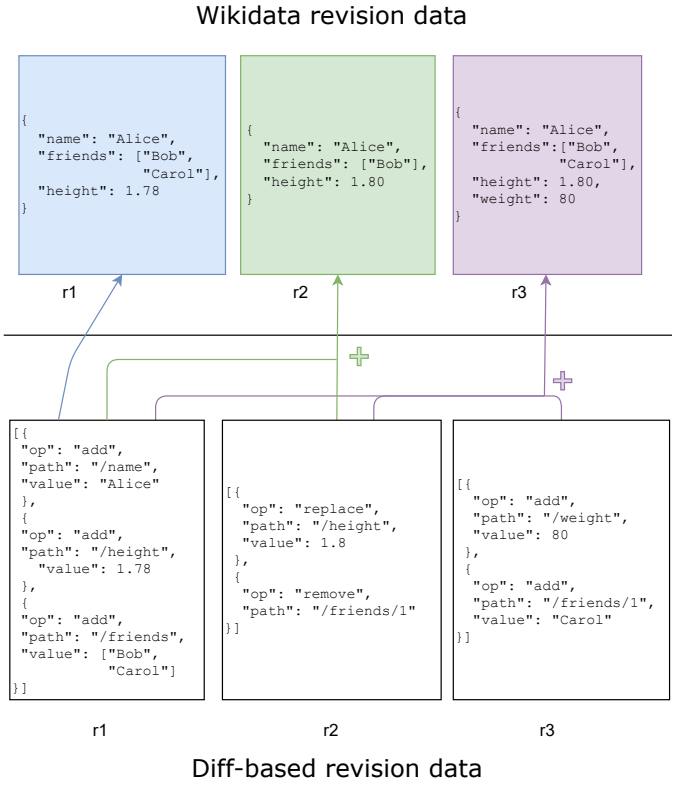


Figure 1: Example of revision decompositions into JSON Patch. At the top of the figure we show the simplified content of a Wikidata entity at each revision r. At the bottom of the figure we show the decomposition of this entity into diffs in JSON format.

[9] Available at `https://dumps.wikimedia.org/wikidatawiki/`

[10] `https://datatracker.ietf.org/doc/html/rfc6902`

Table 2: Statistics of the edits dataset used in the experiments

Dataset size:	401GB
Number of entities:	9,709,099
Number of revisions:	443,116,607
Number of operations:	986,678,861

In this process, the following revision data was extracted:

- **id**: ID of the Wikidata revision.
- **parent_id**: ID of the previous revision affecting the entity.
- **entity_id**: ID of the entity altered by the revision.
- **timestamp**: Date of the revision, following the ISO 8601 format (e.g., +2019-05-27T09:31:10Z).
- **username**: Name of the user that made the revision.
- **comment**: Comments of the revision, if they exist.
- **entity_diff**: Diff of the revision, following the JSON Patch format previously explained.

3.3. Indexing of data

The revision data previously fetched was then indexed into a *MongoDB* database for further exploration. Table 2 shows the main statistics of the revisions dataset that was indexed.

A custom database architecture was defined in order to optimize the performance of queries needed in the following experiments. The database was split into two main collections: one for **entities** and another one for **revisions**. The former contains the final complete JSON content of each entity, while the latter contains the JSON diffs of each entity after each revision. Although the final content of an entity can be obtained by applying all of its revision diffs, this architecture allows for a performant exploration of the revision data and also the current content of each entity.

This dataset has been released in order to promote its exploration by other researchers [15].

4. Wikidata edits analysis

In this section we explore the edits dataset to exploit the dynamic aspects of Wikidata. While Wikidata is usually explored from a static point of view (i.e., having into account just the state of the knowledge graph at a point in time t), having information about its changes lets us understand the community's approach when building the knowledge graph in fine-grained detail.

4.1. Life cycle of an entity

Figure 2 shows a graph of transitions from one edit operation applied to an entity to the next one. Edge weights represent the number of transitions from one operation to another, while node size represents the total for each state –with higher weights representing a larger count. These transition and state counts were accumulated across every entity in the dataset. In order to better illustrate the most common transitions and reduce noise from the graph, those transitions with a count lower than 10% of the total count of outgoing transitions from a state were removed.

We can see how the central operation in this graph is adding a statement group to an entity, followed by adding references to a claim. These are the most prominent operations applied to entities in Wikidata. There is also a group of three operations in the top right corner of the graph, which represents adding single values to the entity: either a single statement, reference, or qualifier. Both behaviors are expected, since a common practice in Wikidata is to add appropriate references and qualifiers to a statement that has been added.

In general, replacements or removals are not as common as addition operations. However, these operations are usually connected between them (i.e., a replacement or removal operation is usually followed by another replacement or removal). While adding a statement may not imply adding a reference or qualifier to it, removing a statement implies the removal of its qualifiers and references. Therefore, some connections between a removal state and the next one are expected.

Finally, changing the rank or order of statements, references, or qualifiers is also an infrequent operation within Wikidata.

4.2. Edit information

Operations applied to the most important classes. Figure 3 shows the top 10 classes with the highest average number of operations performed on its instances. These frequencies are split into additions, removals, and replacements. We can see how in the classes that have a higher operation count replacements are the most prominent operation (over additions and removals). As the number of operations keeps decreasing, replacements are less common and additions become the most common operation. The ratio between additions and removals can be higher or smaller depending on the class, which could reveal of the difficulty of reaching a consensus on the content of its instances.

Another interesting insight from this chart is that some of the most important classes based on their ClassRank score do not have a high number of average operations per instance. In particular, the 2 most important classes (human and taxon) do not appear in the top 10 classes with the most operations. One possible explanation for this could be that instances of these classes do not have as much information to be added as instances of other classes.

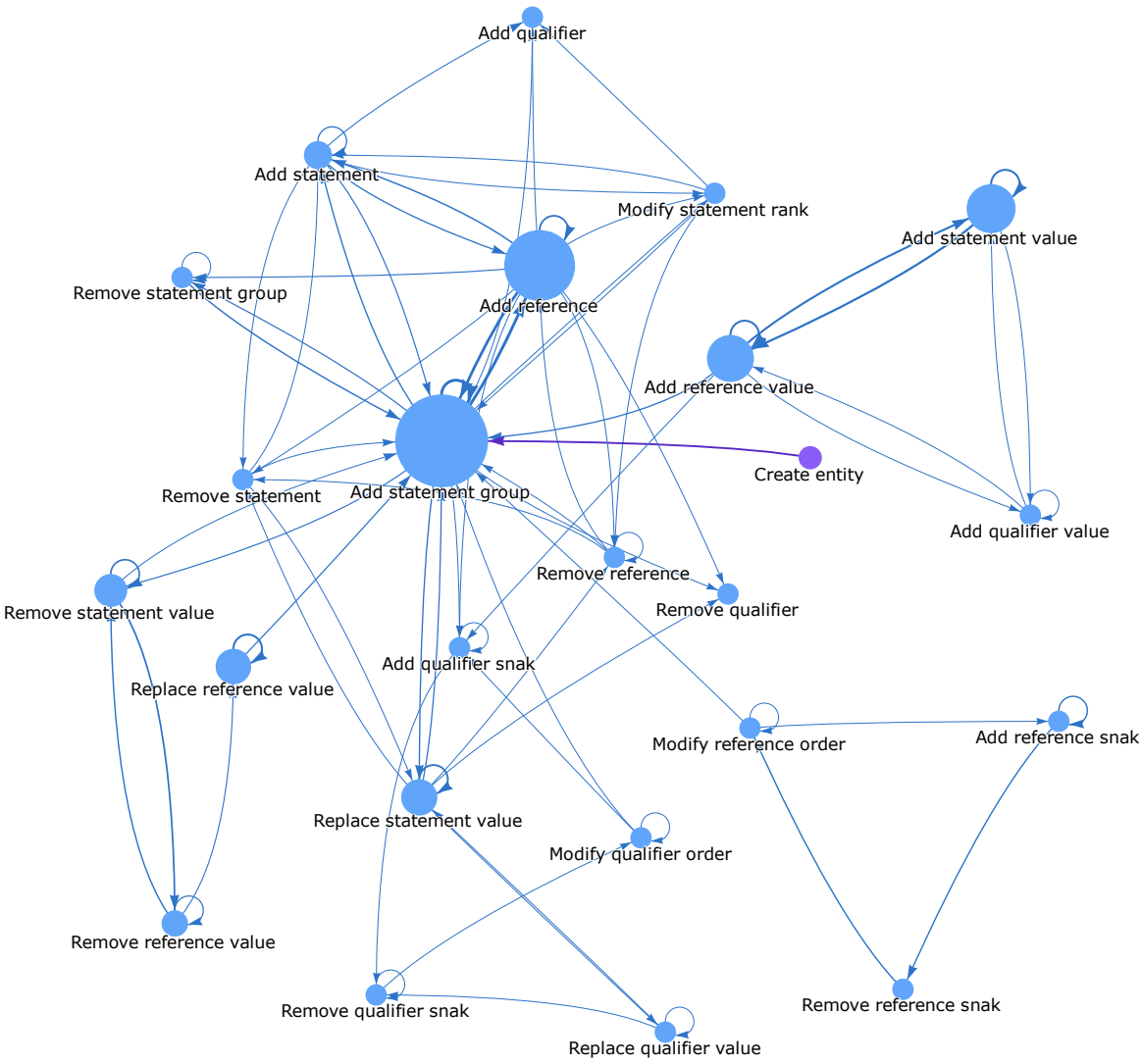


Figure 2: Most common transitions from one edit operation to another in the entities lifetime. The purple circle represents the start in the life cycle of an entity, and blue circles represent different operations applied to the entity during its life cycle.

Another possibility could be that these classes contain a high number of instances that are not edited as often, so the average number of operations is not as high as in other classes.

Most frequently removed properties of a class. Figure 4 shows the most removed properties of 4 classes from the dataset: *human*, *taxon*, *sovereign state*, and *big city*. Properties with the name '*[deleted property](PXX)*' refer to properties that were removed from Wikidata, for which the original label cannot be retrieved[11].

We can see in the figure some differences in the removal behavior for different classes. For some classes, like '*taxon*' and '*big city*', the most removed properties are properties that were directly removed from Wikidata. These properties do not reflect the level of consensus in the class, since they are removed completely from Wikidata at an ontological level. The remaining properties have a lower deletion rate, indicating that the properties used in these classes are overall stable and rarely change.

Other classes, like '*sovereign state*', have a high number of removals across many properties –which are still existing in Wikidata. This means that, in general, instances of that class have a set of properties that either lead to controversy or are evolving naturally along the lifetime of the instance.

Finally, the '*human*' class only has a property with more than 0.3 deletions per instance, which was removed from Wikidata. The rest of the properties have a really low deletion rate. This follows the same trend that we have seen when analyzing the number of operations per class, showing that although the human class is the most important one in Wikidata –based on ClassRank– it does not have a high average number of operations per instance.

A deep understanding of these results across every class could be beneficial to analyze which properties are more controversial and which ones do not experience as many deletions. This information could be used by recommendation systems to aid editors by signaling those properties that are more controversial and need special consideration when used.

4.3. Conflict in Wikidata

Our last set of analyses of the edit history dataset revolves around the idea of divisiveness in Wikidata. We have analyzed the edits dataset to detect cases where an *edit war* between users occurs when editing an entity[12]. In our case we have defined an *edit war* as a succession of edits in which a value for a property is added, the value is removed or replaced by another different value in a later revision, and finally the original value is added again to the entity. Certainly, such a definition does not only cover the case of a conflict within the community on the value that an entity should have for a given property, but also acts of vandalism that are eventually reverted.

Figure 5 shows the properties with the most amount of edit wars in the dataset. The property with the highest edit war count was '*sex or gender*', with a notable difference when compared to other properties. Typing entities with the '*instance of*' property is the next case with the highest amount of edit wars. The remaining properties are more domain-specific and have a considerably lower number of edit wars when compared to the top 2.

In figure 6 we can see the classes with the highest mean amount of edit wars across its instances. Most edit wars occur in classes related to territories, with only some exceptions not belonging to that domain (e.g., chemical element, natural language...). This follows the same pattern we have seen previously, with the '*sovereign state*' class having a high number of deletions and replacements of property values.

5. Type prediction leveraging edit data

In this section, we explore a potential sub-field within knowledge graph refinement where this edit history data could be exploited: type prediction in knowledge graphs. More specifically, our objective is predicting possible values of the *instance of* (*P31*) property of a given entity. In the following sections, we will explore our approach and the results obtained.

5.1. Proposed approaches

In this paper we analyze the impact of leveraging edit history information in knowledge graph embedding models. We have decided to employ these models since they are currently used in type prediction and other knowledge graph refinement tasks.

Since these models usually employ a *'static'* version of a graph (i.e., without including any information about the editions that the graph underwent until reaching its final version), our objective is to include information from the edit history of the graph to obtain an improvement in their performance. To do so, we propose the following two techniques in our experiments: using the removed triples from the graph to enrich negative sampling methods, and labeling each triple that was added or removed from the graph to train a supervised model.

5.1.1. Negative sampling enriched by edit history information

Knowledge graph embedding models perform a process called *negative sampling* when generating the embeddings, where negative triples –corruptions– are generated. Those corruptions are used during the training process of the embeddings as examples of negative triples. This generation

[11] The process of creating and removing properties from Wikidata is more complex than creating entities or adding statements to them. It can only be done by users who have the '*Property Creator*' role, and involves creating a formal proposal and reaching a consensus over the proposal.

[12] The interested reader should consult [16] for more information on edit wars, albeit in Wikipedia, not in Wikidata.

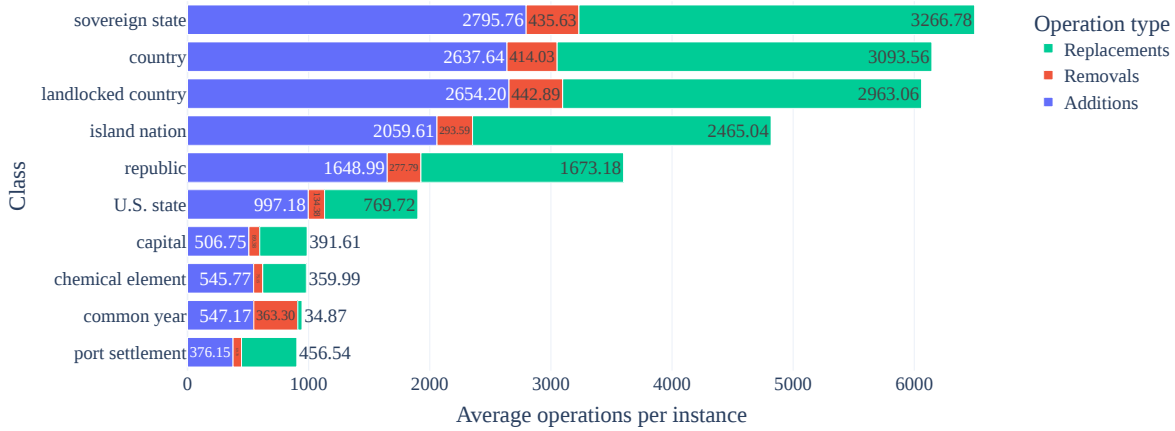


Figure 3: Top 10 classes with the most number of operations performed on its instances.

of corruptions is usually carried out randomly, under the premise that the total space of possible triples ($|\mathcal{S}\times\mathcal{P}\times\mathcal{O}|$) is much greater than the one of true triples, and therefore the possibility of introducing false negatives is small.

However, in the revision history of an entity we have potentially useful information to determine which corruptions could be generated: triples that the community had considered invalid sometime along the life cycle of a given entity. We can draw an analogy of this idea to the task of training an automatic spell checker. One option could be to artificially generate typos on the set of training words by randomly replacing, removing, or adding one character to the word (basic negative sampling approach). However, a more optimal approach could be to also use as a training sample real typos made by humans on these words when writing on their devices (edit history based negative sampling approach).

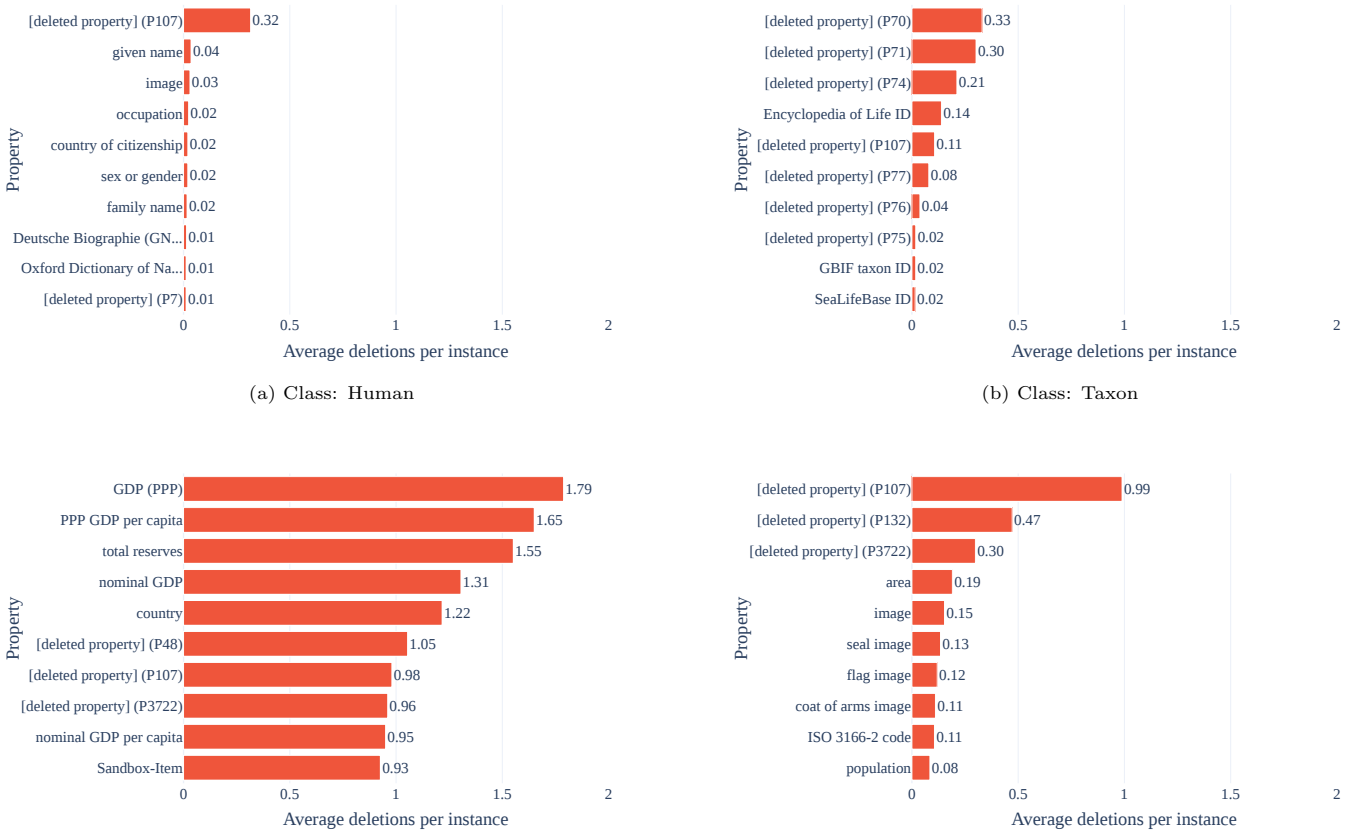
Therefore, we propose the following set of negative sampling methods based on the edit history of a knowledge graph:

- **Edit history negative sampling**. We start by selecting the set of negative triples of each triple that needs to be corrupted. This set of negative triples is obtained from all the removal operations in the knowledge graph that share the same subject and predicate of the given positive triple. For example, if we want to generate corruptions for the triple '`DouglasAdams occupation writer`', we would look into the edit history of the graph for triples that were removed having '`DouglasAdams`' as subject and '`occupation`' as predicate[13]. Afterward, the set of negative triples is shuffled, and the first n elements are selected as the generated corruptions. If the set of negative triples contains less than n elements, the remaining corruptions are generated randomly, following the traditional process. A variation of this technique consists of generating the set of negative triples without including those triples that were the subject of edit wars. In the following experiments, we will refer to models using this type of negative sampling techniques with the terms **'edit hist'** y **'edit hist (no edit wars)'**, with the former including edit wars in the set of negative triples while the later omits them.

- **Inverse edit history negative sampling**. This method is based on the opposite hypothesis with respect to the previous one: triples that were removed by the community represent a conflict of opinion about the value of a property, and they do not entail that its value is necessarily invalid. Therefore, these removed triples should not be added to the corruptions of the model since they are triples that could be considered valid or, at least, partially related to the entity that is being corrupted. For this reason, although we start from the same set of negative triples explained previously, in this case the objective is to avoid the generation of any corruption that belongs to that set. Corruptions are generated randomly, following the traditional approach, but if the generated corruption belongs to the set of negative triples the corruption is generated again randomly until it does not belong to it.

The pseudo-code of the corruption fetching process and both negative sampling approaches is shown in algorithm 1 and algorithm 2 respectively.

[13] In these examples the corruption of objects of each triple is shown. The corruption of subjects is analogous, where we instead select as negative triples those that share the same predicate and object of the positive triple.



(a) Class: Human

(b) Class: Taxon

(c) Class: Sovereign State

(d) Class: Big City

Figure 4: Most deleted properties of each class.

Algorithm 1: Fetching of corruptions from the edit history pseudo-code

Input:

$t = (s, p, o)$ Triple that needs to be corrupted.
$Op^- = \{t_1, t_2 ... t_m\}$ Set of removal operations of the knowledge graph.
$omit\_edit\_wars \in \{True, False\}$ Whether edit wars should be omitted or not.

Variables:

$F \subset Op^-$ Set of all filtered corruptions.

Output:

$T_{neg} \subset F$ Set of negative samples of t.

begin

$F \longleftarrow \emptyset$
for $t' = (s', p', o') \in Op^-$ **do**
  if $omit\_edit\_wars = False$ *or not* $HasEditWars(t')$ **then**
    $F \longleftarrow F \cup \{t'\}$
$T_{neg} \longleftarrow \{(s', p', o') \in F : s' = s, p' = p\}$
return T_{neg}

Algorithm 2: Negative sampling methods pseudo-code

Input:

$n \in \mathbb{N}$: Number of negative samples to be generated per triple.
Op^- Set of removal operations of the knowledge graph.
$T = \{t_1, t_2, ..., t_m\}$ Set of triples to be corrupted.
$omit\_edit\_wars \in \{True, False\}$ Whether edit wars should be omitted or not.

Variables:

T_{neg} Set of negative samples of triple t.

Output:

$C_{out} \subset (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{B})$ Generated corruptions.

Function *InverseNegativeSampling(T, Op^-, $omit\_edit\_wars$, n)*:

$C_{out} \longleftarrow \emptyset$
for $t = (s, p, o) \in T$ **do**
  $T_{neg} \longleftarrow FetchCorruptions(Op^-, t, omit\_edit\_wars)$
  for $i = 1 : n$ **do**
    do
      $t' \longleftarrow RandomCorruption(t)$
    while $t' \notin T_{neg}$
    $C_{out} \longleftarrow C_{out} \cup t'$
return C_{out}

Function *EditHistoryNegativeSampling(T, Op^-, $omit\_edit\_wars$, n)*:

$C_{out} \longleftarrow \emptyset$
for $t = (s, p, o) \in T$ **do**
  $T_{neg} \longleftarrow FetchCorruptions(Op^-, t, omit\_edit\_wars)$
  while $|T_{neg}| < n$ **do**
    $T_{neg} \longleftarrow T_{neg} \cup RandomCorruption(t)$
  $T_{neg} \longleftarrow Shuffle(T_{neg})$
  $C_{out} \longleftarrow C_{out} \cup \{t_1, t_2 ... t_n : t_i \in T_{neg}\}$
return C_{out}

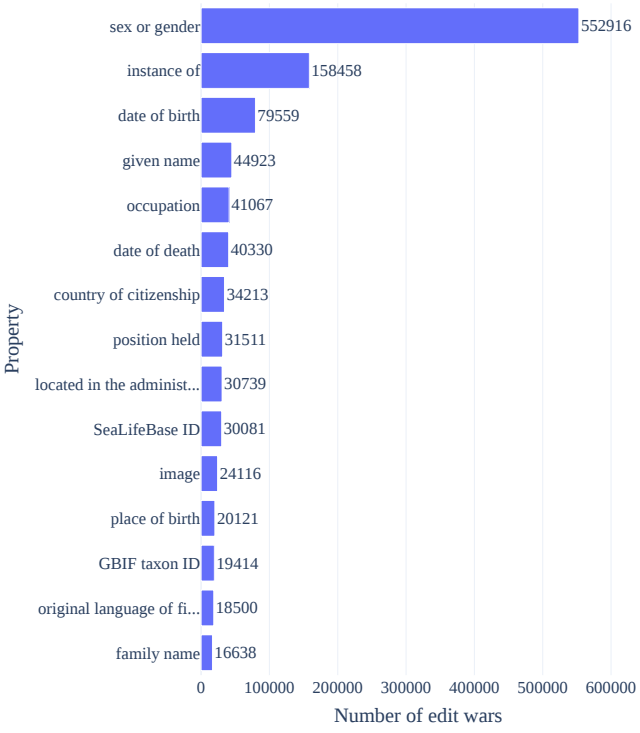


Figure 5: Top 15 properties with the highest amount of edit wars.

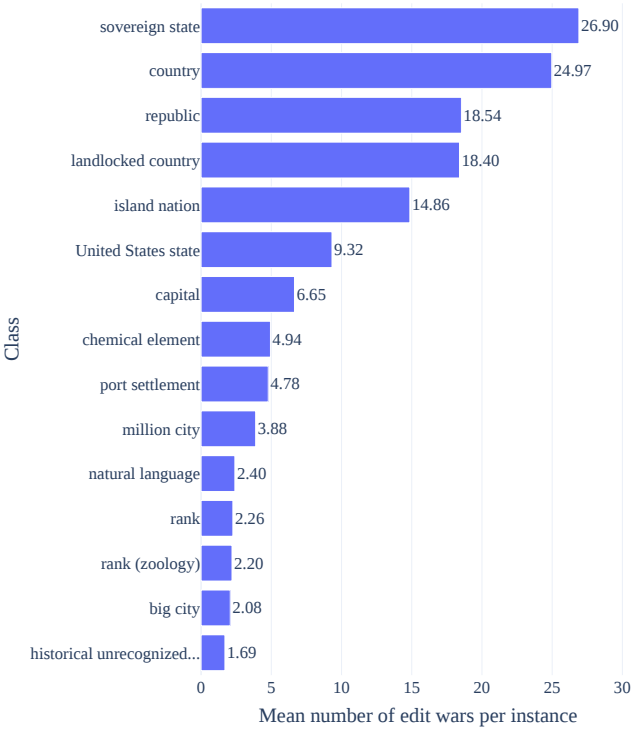


Figure 6: Top 15 classes with the highest amount of edit wars.

5.1.2. *Transformation of non-supervised task to a supervised one*

While the generation of knowledge graph embeddings is a non-supervised technique where it is not necessary to label any triple from the knowledge graph, having information about the edit history of the graph allows us to convert this task into a supervised one.

To be more precise, from the set of addition operations Op^+ and the set of removal operations Op^-, we can assign a negative label $y = 0$ to each triple $t \in Op^-$, while assigning a positive label $y = 1$ to each triple $t \in Op^+$. With this approach, we can train a predictive model from the embeddings representation of each triple and their corresponding label obtained from the edit history of the knowledge graph. In our experiments the input of the classifiers will be the concatenation of the subject, predicate, and object embeddings. In order to obtain a ranking of triples for evaluation purposes, we have ordered the results by the log probability given by the classifier of the results being positive.

5.2. *Dataset*

Knowledge graph embedding models work on a graph version of a dataset, while our dataset indexed on MongoDB that has been previously explored is stored in JSON format. Therefore, to run these models it was necessary to transform that data into RDF.

The first step of that process is to transform the 'static' JSON content of each entity to RDF. To do so, we transformed just the *simple statements* of every entity, ignoring complex values like qualifiers and references. Since the JSON dataset indexed in MongoDB is composed of millions of triples, a subset of this data was selected to be converted to RDF. This subset has 300 randomly selected instances from each class indexed in MongoDB, resulting in 390.000 triples in the final graph.

We also created an additional RDF dataset containing the 'dynamic' information of each entity (i.e., the edit history information). This dataset represents each revision made to an entity, plus the atomic operations that compose the revision. It is possible to build the static RDF graph by applying each operation on the dynamic version, following the same principles as with the JSON dataset.

Finally, both datasets (dynamic and static) were split into training, validation, and test sets. Every entity is maintained on each set when making the divisions, but in the training dataset we include the state of every entity until a time step t_i, in the validation set their new state from a time step t_i until t_{i+n}, and in the test set its new state since time step t_{i+n} until now. In our particular case, the training set includes the first 70% revisions of each entity (in chronological order), the validation set includes an additional 15% (between the 70% and 85% revisions), and the test set contains the last 15% of revisions.

5.3. Training

Unsupervised models. We have selected 3 knowledge graph embedding models to evaluate the impact of our proposed negative sampling techniques: RotatE [17], TransE [18], and MuRE [19]. Those models were selected based on the large-scale benchmarks done by Ali et al. under the PyKEEN framework [20]. We have chosen those with a good performance under the Yago dataset[14] and that can scale to large dataset sizes without requiring large computational resources.

Hyperparameters were optimized for each model using the training and validation sets. Since the total space of parameter combinations was too big to be completely covered, a random search was made to find the best parameters. Each model parameters were optimized for 80h, for a total of 240h (10 days). Table 3 shows the final set of optimized parameters found for each model. Regarding parameters specific to each model, TransE had an optimized parameter $scoring\_fct\_norm = 2$ and MuRE's best value for the l_p norm parameter was 2.

To make the most out of the available data in the dataset, each combination of model and negative sampler was then trained on the union of both the training and validation sets. That is, the final training set of each model contains the 85% first revisions of each entity, while the last 15% of revisions are reserved for the test set.

Supervised model. For the supervised version of this task we have first trained RDF2Vec embeddings [21] on the dataset, and fed the embeddings to a Random Forest classifier with the labeled data.

The same optimization and training process done with the unsupervised models was followed with the supervised approach. In this case, we have also optimized the parameters of both the RDF2Vec and the Random Forest models, performing a random search over a space of parameters for each model. The final set of parameters found for the RDF2Vec model was a vector size of 50, with 50 training epochs, a maximum depth of 5, and 50 walks generated. Regarding the random forest classifier, the set of parameters found was 75 for the number of trees, a minimum of 2 samples to perform a split, and the *gini* criterion to perform splits.

5.4. Evaluation

To measure the performance of each model in the test set we have used the following metrics: **mean rank (MR)**, **mean reciprocal rank (MRR)**, and **hits@1**, **5**, and **10**. We selected those entities which had a new class added to the test set (i.e., a new value for the *instance of* property), and run each model to get the ranked list of class suggestions for the entity. We have followed the 'filtered setting' proposed by Bordes et al. in the evaluation process by removing those triples suggested by the model which had already appeared in the train set (*true triples*) [18]. The final test sample is made of 4193 triples, which are the triples from the test dataset containing the *instance of (P31)* property.

[14] Due to it being the dataset more closely related to the one used in this paper.

5.5. Results

5.5.1. Negative sampling techniques

The results obtained for each combination of unsupervised model and negative sampling technique are shown in table 4. Results for the *hits@5* measure across every model combination are graphically shown in figure 7.

We performed McNemar tests over each combination of models to prove whether there are statistically significant differences ($pvalue \leq 0.01$) between the results obtained [22]. To build the contingency table we measured the accuracy (hits@1) of each model being compared, storing those cases when both models guessed correctly the new class of an entity, those cases when both models failed to guess the class, and the cases when one of the models guessed the class correctly while the other did not. Therefore, each comparison will measure whether both models have a similar proportion of errors in the test set. Since we are interested in the differences between each negative sampler, we only performed comparisons between the different negative samplers of each model type.

Results of running the McNemar test over each pair of negative samplers show the following statistically significant differences for the RotatE model: *basic* against *edits* [$\chi^2(1) = 15.00$, $pvalue = 1 \cdot 10^{-4}$]; *basic* against *edits (no wars)* [$\chi^2(1) = 79.27$, $pvalue = 5.4 \cdot 10^{-19}$]; *basic* against *inverse* [$\chi^2(1) = 144.27$, $pvalue = 3.1 \cdot 10^{-33}$]; *edits* against *edits (no wars)* [$\chi^2(1) = 39.75$, $pvalue = 2.8 \cdot 10^{-10}$]; *edits* against *inverse* [$\chi^2(1) = 72.20$, $pvalue = 1.9 \cdot 10^{-17}$].

With regards to the TransE model, we obtain the following statistically significant differences: *basic* against *edits* [$\chi^2(1) = 72.90$, $pvalue = 1.3 \cdot 10^{-17}$]; *basic* against *edits (no wars)* [$\chi^2(1) = 25.44$, $pvalue = 4.5 \cdot 10^{-7}$]; *basic* against *inverse* [$\chi^2(1) = 54.37$, $pvalue = 1.6 \cdot 10^{-13}$]; *edits* against *edits (no wars)* [$\chi^2(1) = 16.66$, $pvalue = 4.4 \cdot 10^{-5}$];.

Regarding the MuRE model, there is a statistically significant difference between every negative sampler, with the following detailed results: *basic* against *edits* [$\chi^2(1) = 274.69$, $pvalue = 1 \cdot 10^{-61}$]; *basic* against *edits (no wars)* [$\chi^2(1) = 206.86$, $pvalue = 6.6 \cdot 10^{-47}$]; *basic* against *inverse* [$\chi^2(1) = 20.48$, $pvalue = 6.0 \cdot 10^{-6}$]; *edits* against *edits (no wars)* [$\chi^2(1) = 12.69$, $pvalue = 3.6 \cdot 10^{-4}$]; *edits* against *inverse* [$\chi^2(1) = 393.26$, $pvalue = 1.6 \cdot 10^{-87}$]; *edits (no wars)* against *inverse* [$\chi^2(1) = 307.65$, $pvalue = 7.1 \cdot 10^{-69}$].

The remaining combinations of negative samplers do not show statistically significant differences ($pvalue > 0.01$), with the following detailed results: RotatE *edits (no wars)* and RotatE *inverse* [$\chi^2(1) = 6.01$, $pvalue = 0.014$]; TransE

Table 3: Optimized hyperparameters of each unsupervised model

Model	Embedding dim.	Num. epochs	Batch size	Learning rate	Num. negatives
RotatE	768	13	64	0.009	5
TransE	64	10	521	0.024	7
MuRE	150	21	256	0.088	28

Table 4: Evaluation results of unsupervised models. '*basic*' represents the basic negative sampler currently used in knowledge graph embedding models; '*edits*' and '*edits (no w.)*' refer to the proposed negative sampling techniques that generate corrupt triples from removals of the edit history, considering or omitting edit wars respectively; and '*inv.*' refers to the proposed inverse edit history negative sampling, where triples that were removed from the knowledge graph cannot be generated as corruptions.

	RotatE				TransE				MuRE			
	basic	edits	edits (no w.)	inv.	basic	edits	edits (no w.)	inv.	basic	edits	edits (no w.)	inv.
MR	10819	21268	23577	**7197**	3204	**2385**	2756	2595	527	4650	3385	**447**
MRR	0.177	0.175	0.211	**0.260**	**0.091**	0.043	0.055	0.046	0.204	0.063	0.085	**0.237**
hits@1	0.086	0.109	0.146	**0.163**	**0.050**	0.015	0.028	0.020	0.115	0.018	0.031	**0.144**
hits@5	0.300	0.241	0.280	**0.382**	**0.121**	0.055	0.072	0.054	0.294	0.090	0.122	**0.330**
hits@10	0.382	0.308	0.339	**0.445**	**0.164**	0.091	0.097	0.083	0.385	0.148	0.210	**0.422**

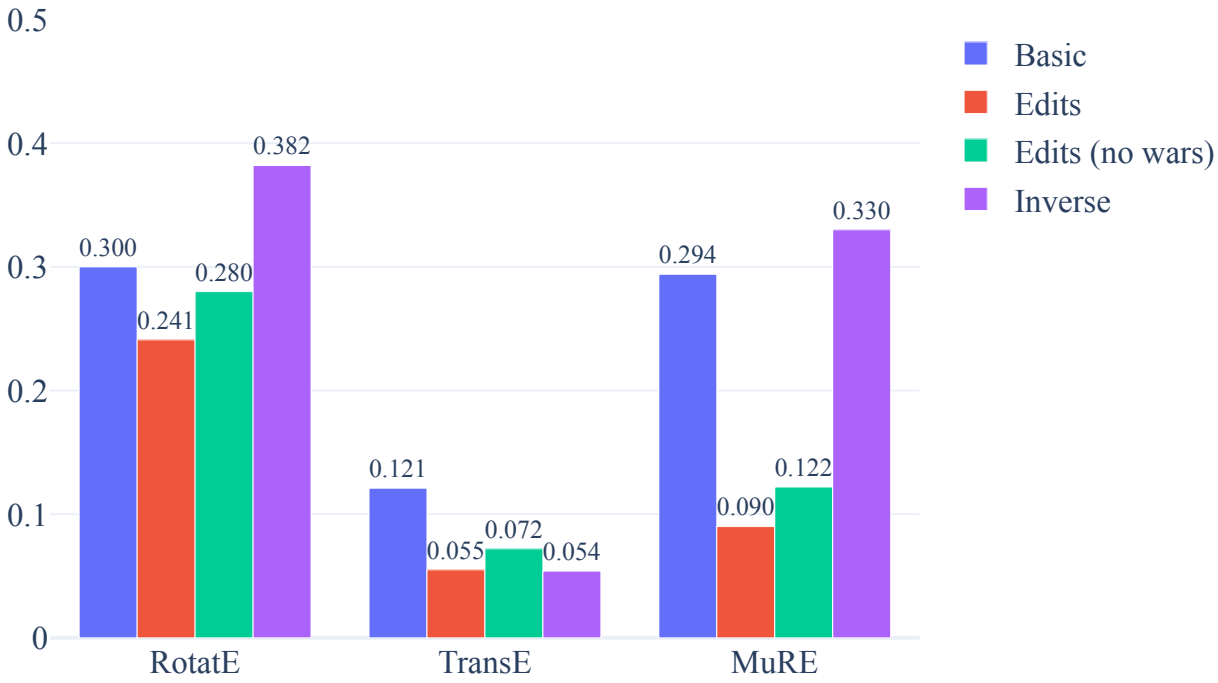


Figure 7: Hits@5 results of each unsupervised model.

edits and TransE *inverse* [$\chi^2(1) = 2.58$, $pvalue = 0.10$]; TransE *edits (no wars)* and TransE *inverse* [$\chi^2(1) = 5.80$, $pvalue = 0.016$].

In order to report the impact of our proposed negative samplers we also follow the evaluation reporting approach proposed by Sparck-Jones [23]: performance differences are regarded as *non-relevant* if they are lower than 5%; as *noticeable* if they are between $5 - 10\%$; and as *material* if they are greater than 10%.

We can observe from the results that the use of the inverse edit history negative sampling technique has a positive effect on both the RotatE and MuRE models across every measure, having a *material* and *noticeable* impact respectively. However, with the TransE model this is not the case, with the performance of the basic negative sampler improving that of the proposed negative samplers across almost every measure. The only exception is the mean rank measure, where the basic negative sampler has on average a 24.68% higher mean rank score than the proposed approaches. It is important to note that the TransE model results are in general considerably worse than the RotatE and MuRE ones, regardless of the negative sampler being used. This could be due to not finding an optimal set of parameters for the model, or to the model having a worse performance in this dataset with respect to the other alternatives. In general, using the proposed inverse edit history negative sampler seems to influence positively in the results, improving the performance of 2 out of the 3 models being evaluated and obtaining the 2 best results in terms of MRR and hits@k.

On the other hand, an improvement over the basic approach is not as clear in the case of the edit history negative samplers. If we consider the RotatE model, both edit history negative samplers still perform better than the basic approach in terms of hits@1, but the improvements in performance are not as high as in the case of the inverse negative sampler. However, in the remaining models the performance of the edit history negative samplers is worse than the basic sampling approach across almost every metric. This could be attributed to the fact that, in some cases, a value that was removed from an entity may actually be true –or at least, partially related to the domain of the entity being affected.

We are going to illustrate this with a real example from the dataset: the entity representing *MAN Truck & Bus*[15] was initially an instance of '*company*'[16] in Wikidata, but this was later removed and replaced by the '*business*'[17] class. Although there are some legal details that differentiate both terms, they are still really close from a conceptual point of view, especially for Wikidata contributors without expertise in business law. In cases like the one mentioned above, generating the removed triple (`MAN instanceof company`) as a corruption could add noise to the model. This could also be a reason for the good performance of the inverse negative sampling approach, since it avoids those cases.

It is also crucial to understand that there is a random component to the edit history negative sampling methods. Since we are fetching n random negative triples for each triple that needs to be corrupted, in some cases we may be fetching true negative triples (e.g., an edit that removed an act of vandalism) while in other cases we may be getting negative triples with a close relation to the entity conceptually (like the MAN Truck & Bus example mentioned above) or even a removal of a completely valid triple. This can explain some of the variability in the performance of the edit history negative samplers and leads to the need of further analysis which is addressed in the future work section.

Another interesting insight from these results is that omitting edit wars is beneficial for the performance of the edit history negative samplers across every model. This is intuitively expected, since values that cause conflict between the community should not be added –in general– as false triples to the model. These values are more commonly associated with information that is not clear whether an entity should have or not, and therefore are not usually related to incorrect information. Furthermore, although these values were removed in the training set, since they were part of edit wars it is more probable that they will be part of the added triples in the test set. Establishing these triples as false in our negative sampling approach will decrease their suggested ranking in the testing phase, lowering the performance of the model.

5.5.2. Supervised task

Table 5 shows the results obtained by the supervised approach against each unsupervised model with its best negative sampler.

The same process was followed to determine whether there are statistically significant differences between the error rates of each model. Results of the McNemar test showed a statistically significant difference across every combination of models, with the following detailed results: RotatE against TransE [$\chi^2(1) = 15.00$, $pvalue = 1 \cdot 10^{-4}$]; RotatE and MuRE [$\chi^2(1) = 79.27$, $pvalue = 5.4 \cdot 10^{-19}$]; RotatE and RDF2Vec + Random Forest [$\chi^2(1) = 15.00$, $pvalue = 1 \cdot 10^{-4}$]; TransE and MuRE [$\chi^2(1) = 79.27$, $pvalue = 5.4 \cdot 10^{-19}$]; TransE and RDF2Vec + Random Forest [$\chi^2(1) = 79.27$, $pvalue = 5.4 \cdot 10^{-19}$]; MuRE and RDF2Vec + Random Forest [$\chi^2(1) = 79.27$, $pvalue = 5.4 \cdot 10^{-19}$].

As we can see in the results, the supervised approach (RDF2Vec + Random Forest) is the worst approach in terms of mean rank (**MR**) by a considerable amount, having a 34 times larger mean rank than the best approach (MuRE). This larger decrease in mean rank score with respect to other metrics is expected, since the supervised approach converts the type prediction task to a binary classification problem. That is, this model tries to maximize

[15] `https://www.wikidata.org/wiki/Q708667`
[16] `https://www.wikidata.org/wiki/Q783794`
[17] `https://www.wikidata.org/wiki/Q4830453`

the probability of guessing correctly whether an entity is true or false: the ranking of true over false triples does not matter to the model.

With respect to the other metrics, its performance is better than TransE but worse than both RotatE and MuRE. This follows the same trends that were outlined when analyzing the performance of each negative sampler. Using removals from the edit history as false triples to the classifier adds a lot of noise in some cases, decreasing the overall performance of the model.

5.6. Reproducibility

Experiments were executed using Python 3.8. Knowledge graph embedding models were created and trained using the PyKEEN 1.8.0 [24] and PyTorch 1.11.0 [25] libraries. The supervised model was built with pyRDF2Vec 0.2.3 [26], and Scikit-learn 1.0.2 [27]. A random seed of 42 was established to create the RDF dataset and train each model.

The complete code where all the experiments of this work were conducted is also available at `https://github.com/alejgh/wikidata-edithist-refinement`. The RDF datasets with static and dynamic serializations used in the type prediction task are also publicly available [28]. We think that new research opportunities can arise by exploiting the dynamic information of an RDF dataset beyond the negative sampling techniques proposed in this paper, and both datasets could be used to measure the performance of models leveraging the edit history of a knowledge graph.

6. Related Work

6.1. Collaboration dynamics of Wikidata

Several works have tried to better understand the collaboration and edition dynamics of Wikidata. T. Steiner was one of the first authors exploring this field, by implementing a tool that provided real-time statistics about edits made in both Wikipedia and Wikidata [29]. That author performed a quantitative analysis of the editions made to Wikidata across 3 days in 2013, focusing on the impact that bots had on the knowledge graph. Their analysis shows that –during that time frame– most edits in Wikidata were made by bots (around 88%).

While the quantitative analysis of Steiner provided a better understanding of the editor distribution in Wikidata, a qualitative analysis was still needed to understand which aspects of Wikidata were edited by each contributor, and therefore get a better view of their behavior within the Wikidata ecosystem. This is one of the main contributions of the study conducted by Müller-Birn et al. in 2015 [30]. Those authors performed clustering on edit statistics computed across 1 month time frames, and those clusters were used to detect different editing roles in Wikidata (e.g., property editor, domain expert, instance creator...). Those editing roles correspond to the different collaboration patterns that exist in Wikidata.

Another point of interest within the field is understanding the path that a new editor follows to become an established contributor within Wikidata. Piscopo et al. detected that Wikidata has a low entry barrier, allowing newcomers to start making simple statement contributions right away [31]. However, as contributors become more expert around the Wikidata ecosystem they transition from performing editions through the interface to using more advanced automatic tools. Understanding at an early stage which editors may be more engaged with editing in Wikidata was the goal of Sarasua et al. [32]. They monitored the way users participated in Wikidata for 4 years, and developed predictive models to estimate the edit lifetime of a contributor reaching an F1-score of 0.9.

More recent works have shifted to understanding how these collaboration dynamics affect the overall quality of the knowledge graph. Piscopo et al., analyzed several edit metrics of 5,000 Wikidata items to understand these relationships [33]. They detected that the interaction between human and bot users is crucial to ensure the quality of Wikidata items, and that anonymous edits are instead detrimental to the quality (mainly due to vandalism). In their follow-up work, they tried to detect a correlation between a set of defined quality metrics and editing roles detected with a clustering approach [34]. Results showed that Wikidata is unevenly distributed at an ontological level, with some domains having a lot of sub-classes automatically created by bots.

6.2. Type prediction in knowledge graphs

Regarding works on type prediction tasks, SDType was one of the first tools made to add missing type statements in knowledge graphs [11] –specifically, in DBPedia. SDType works by calculating the statistical distribution of types in the object and subject position of a property and using this data to fill missing types of an entity, outperforming other type prediction approaches at the time. Melo et al. then proposed a new system that exploited the hierarchical nature of type systems in knowledge graphs by using a hierarchical classifier [35]. This system was tested on several knowledge graphs, including Wikidata, outperforming SDType and other alternatives while also scaling better than other multilabel classifiers.

The use of knowledge graph embeddings for the task of type prediction has been explored recently. Kejriwal et al. proposed a method to create class embeddings based on a series of pre-computed entity embeddings [36]. Class embeddings are closer to related instances and similar classes in the vector space. Those authors then proposed the use of cosine similarity between the embedding of an entity and the class embeddings to recommend the typing of the entity. Other approaches try to generate fine-grained type predictions using entity embeddings in both super-

Table 5: Evaluation results of each best unsupervised model and the supervised model.

	RotatE (inv.)	TransE (basic)	MuRE (inv.)	RDF2Vec + Random Forest
MR	7191	2385	**447**	15752
MRR	**0.260**	0.091	0.237	0.124
hits@1	**0.163**	0.050	0.144	0.062
hits@5	**0.382**	0.121	0.330	0.170
hits@10	**0.445**	0.164	0.422	0.249

vised and unsupervised approaches, obtaining better results than SDType in two DBPedia datasets [37].

Ridle is a system that leverages neural networks and a custom knowledge graph representation to improve existing type prediction approaches [38]. The proposed representation of each entity is built from its relations, under the idea that some sets of properties are more likely to appear as relations in the instances of a class than in others. Ridle outperforms SDType and other embedding-based approaches across multiple datasets, including Wikidata.

6.3. Negative sampling methods

Negative sampling methods have been shown to be an important step in the performance of knowledge graph embedding models. Kotnis et al. proposed two new embedding based negative sampling methods, comparing their performance to random approaches [39]. Their results indicated that the performance of each negative sampling method depended on the characteristics of the dataset being used, with no approach being the best across all datasets.

Using generative adversarial networks (GANs) to produce negative samplers has also been introduced recently, showing promising results by improving the performance of knowledge graph embeddings under different settings [40, 41]. NSCaching is a method that improves the efficiency of GAN based methods by incorporating a cache of high-quality negative triples, outperforming state-of-the-art negative sampling methods [42, 43].

6.4. Leveraging the edit history of knowledge graphs

Exploiting edit information of knowledge graphs for several tasks has been increasingly studied recently. Initial work tried to offer better ways to access and query edits from Wikidata. The Wikidata edit history query service was one of the first available tools, providing a SPARQL endpoint to query edit information made to Wikidata until July 1st of 2018 [44]. During the time of experimentation of this paper (November 2021 to March 2022) that service was down, but at the time of this writing it is back online.

Wikidated is a dataset that contains the entire edit history of Wikidata encoded as a set of triples additions and deletions [45]. It provides a Python API for exploring and interacting with the dataset without needing to know the underlying data model being used. This dataset could not be used in our research since it is still undergoing active development.

Regarding the application of edit history information to knowledge graph refinement tasks, AlGhamdi et al. proposed a Wikidata recommendation system that leverages this information [46]. Their system summarizes the editing activities of the user and combines this with entity representations to recommend items that the user could edit next. Results are promising and lay the ground for future work in Wikidata recommendation systems.

Finally, Tanon et al. proposed a system that automatically mines knowledge graph correction rules from its edit history [47]. Their system starts from a set of constraints violations from the knowledge graph, and mines the edit history to find edits that solved those violations in the past, which are then returned as a set of possible correction rules for the violation. They run a set of experiments in Wikidata, showing significant improvements over baselines. In their follow up work, they proposed a refinement of their previous system by using neural networks to predict the correction rules of a given constraint [48]. That neural network receives as input a vector representation of the constraint, the triples that have been violated, and facts about the entities. Results over the Wikidata dataset showed significant improvements over both the previous rule-based approach and the baselines.

7. Conclusions and future work

This paper provides the first exploration of the use of edit history information in knowledge graph refinement and, specifically, type prediction tasks. We have built and provided a JSON dataset containing the edit history information of every Wikidata instance from the top 100 most important Wikidata classes based on their ClassRank score. This edit history information has been explored, making a special emphasis on its potential influence in knowledge graph refinement tasks. Finally, we have proposed two new approaches to improve knowledge graph embeddings in type prediction tasks: the use of negative samplers that generate corruptions based on the edit history of the graph and training a supervised model by labeling triples based on the edit history information. These

approaches have been evaluated on a RDF dataset against currently used approaches in the field.

The main findings of our work are:

- Instances in Wikidata have different edit behaviors based on the class they belong to. Some classes lead to more controversial statements that are the subject of edit wars, while some other classes are more stable and experience almost no removals or value replacements.
- Using removed triples from the graph as negative samples can improve the performance of a basic negative sampling approach in some cases, but in general it results in a worse performance of the models. On average, a basic negative sampling approach leads to a 123% higher hits@1 score across every model compared to performing negative sampling from removed triples of the graph. Avoiding removed triples that experienced edit wars improves the performance of our proposed edit history based negative sampling approach across every knowledge graph embeddings model. On average, avoiding these triples improves the hits@5 of our proposed approach by 27.18%.
- On the contrary, avoiding the generation of corruptions that belong to the set of removed triples from the graph improves the performance of 2 out of the 3 models, getting the 2 best results across every model and negative sampler combination being evaluated. This approach improves the hits@1 and hits@5 of the basic negative sampler approach by a 89.53% and a 27.33% respectively.

New promising research lines have emerged from this work, among them:

- Reaching a deeper understanding on the differences between edits related to vandalism, a conflict of opinion within the community, or the natural evolution of an entity. Having a method to distinguish between such kinds of edits could be crucial to have a better understanding of the editing behavior in Wikidata and also to improve and create new methods that use this edit information.
- To perform an analysis of the corruptions generated by the negative samplers proposed in this work. This could provide a better understanding of the strengths and weaknesses of each model, while also providing an explanation of which types of edits should be avoided or considered by the negative samplers.
- There are also several optimizations that could be applied to the negative samplers proposed in this work. One of them could be to assign a score to each one of the corruptions generated by the negative samplers, and choosing the top n corruptions based on this score instead of choosing them randomly.
- Performing an incremental evaluation of the performance of each negative sampler approach depending on the size of the dataset, and the number of negative triples from the edit history avoided or added. This evaluation could also include a comparison to more advanced negative sampling approaches (e.g., GAN-based negative samplers).
- Finally, the use of edit history information in other knowledge graph refinement tasks and models could also be explored. For example, one interesting research direction could be the creation of models that work directly with the dynamic version of the RDF dataset provided in this work.

CRediT author statement

Alejandro Gonzalez-Hevia: Methodology, Conceptualization, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing - Original Draft, Visualization, Project Administration. **Daniel Gayo-Avello:** Methodology, Resources, Writing - Review & Editing, Supervision.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] T. Ruan, L. Xue, H. Wang, F. Hu, L. Zhao, J. Ding, Building and exploring an enterprise knowledge graph for investment analysis, in: International semantic web conference, Springer, 2016, pp. 418–436.

[2] M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, D. Sontag, Learning a health knowledge graph from electronic medical records, Scientific reports 7 (1) (2017) 1–11.

[3] A. Nayak, V. Kesri, R. K. Dubey, Knowledge graph based automated generation of test cases in software engineering, in: Proceedings of the 7th ACM IKDD CoDS and 25th COMAD, 2020, pp. 289–295.

[4] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann, Dbpedia-a crystallization point for the web of data, Journal of web semantics 7 (3) (2009) 154–165.

[5] D. Vrandečić, Wikidata: A new platform for collaborative data collection, in: Proceedings of the 21st international conference on world wide web, 2012, pp. 1063–1064.

[6] A. Piscopo, E. Simperl, What we talk about when we talk about wikidata quality: a literature survey, in: Proceedings of the 15th International Symposium on Open Collaboration, 2019, pp. 1–11.

[7] M. Färber, F. Bartscherer, C. Menne, A. Rettinger, Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago, Semantic Web 9 (1) (2018) 77–129.

[8] K. Shenoy, F. Ilievski, D. Garijo, D. Schwabe, P. Szekely, A study of the quality of wikidata, Journal of Web Semantics 72 (2022) 100679.

[9] P. F. Patel-Schneider, Using description logics for rdf constraint checking and closed-world recognition, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[10] E. Prud'hommeaux, J. E. Labra Gayo, H. Solbrig, Shape expressions: an rdf validation and transformation language, in: Proceedings of the 10th International Conference on Semantic Systems, 2014, pp. 32–40.

[11] H. Paulheim, C. Bizer, Improving the quality of linked data using statistical distributions, International Journal on Semantic Web and Information Systems (IJSWIS) 10 (2) (2014) 63–86.

[12] D. Fernández-Álvarez, J. Frey, J. E. Labra Gayo, D. Gayo Avello, S. Hellmann, Approaches to measure class importance in knowledge graphs, Plos one 16 (6) (2021) e0252862.

[13] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web., Tech. rep., Stanford InfoLab (1999).

[14] A. Thalhammer, A. Rettinger, PageRank on Wikipedia: Towards General Importance Scores for Entities, in: The Semantic Web: ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 – June 2, 2016, Revised Selected Papers, Springer International Publishing, Cham, 2016, pp. 227–240. doi:10.1007/978-3-319-47602-5_41.
URL https://dx.doi.org/10.1007/978-3-319-47602-5_41

[15] [dataset] A. Gonzalez-Hevia, Wikidata subset with revision history information [JSON] (Jun. 2022). doi:10.5281/zenodo.6614264.
URL https://doi.org/10.5281/zenodo.6614264

[16] T. Yasseri, R. Sumi, A. Rung, A. Kornai, J. Kertész, Dynamics of conflicts in wikipedia, PloS one 7 (6) (2012) e38869.

[17] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, arXiv preprint arXiv:1902.10197.

[18] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in neural information processing systems 26.

[19] I. Balazevic, C. Allen, T. Hospedales, Multi-relational poincaré graph embeddings, Advances in Neural Information Processing Systems 32.

[20] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, M. Galkin, S. Sharifzadeh, A. Fischer, V. Tresp, J. Lehmann, Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework, IEEE Transactions on Pattern Analysis and Machine Intelligence.

[21] P. Ristoski, H. Paulheim, Rdf2vec: Rdf graph embeddings for data mining, in: International Semantic Web Conference, Springer, 2016, pp. 498–514.

[22] T. G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural computation 10 (7) (1998) 1895–1923.

[23] K. S. Jones, Automatic indexing, Journal of documentation.

[24] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp, J. Lehmann, PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings, Journal of Machine Learning Research 22 (82) (2021) 1–6.
URL http://jmlr.org/papers/v22/20-825.html

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[26] G. Vandewiele, B. Steenwinckel, T. Agozzino, F. Ongenae, pyrdf2vec: A python implementation and extension of rdf2vecdoi:10.48550/ARXIV.2205.02283.
URL https://arxiv.org/abs/2205.02283

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[28] [dataset] A. Gonzalez-Hevia, Wikidata subset with revision history information [RDF] (Jun. 2022). doi:10.5281/zenodo.6613875.
URL https://doi.org/10.5281/zenodo.6613875

[29] T. Steiner, Bots vs. wikipedians, anons vs. logged-ins (redux) a global study of edit activity on wikipedia and wikidata, in: Proceedings of The International Symposium on Open Collaboration, 2014, pp. 1–7.

[30] C. Müller-Birn, B. Karran, J. Lehmann, M. Luczak-Rösch, Peer-production system or collaborative ontology engineering effort: What is wikidata?, in: Proceedings of the 11th International Symposium on Open Collaboration, 2015, pp. 1–10.

[31] A. Piscopo, C. Phethean, E. Simperl, Wikidatians are born: Paths to full participation in a collaborative structured knowledge base.

[32] C. Sarasua, A. Checco, G. Demartini, D. Difallah, M. Feldman, L. Pintscher, The evolution of power and standard wikidata editors: comparing editing behavior over time to predict lifespan and volume of edits, Computer Supported Cooperative Work (CSCW) 28 (5) (2019) 843–882.

[33] A. Piscopo, C. Phethean, E. Simperl, What makes a good collaborative knowledge graph: group composition and quality in wikidata, in: International Conference on Social Informatics, Springer, 2017, pp. 305–322.

[34] A. Piscopo, E. Simperl, Who models the world? collaborative ontology creation and user roles in wikidata, Proceedings of the ACM on Human-Computer Interaction 2 (CSCW) (2018) 1–18.

[35] A. Melo, J. Völker, H. Paulheim, Type prediction in noisy rdf knowledge bases using hierarchical multilabel classification with graph and latent features, International Journal on Artificial Intelligence Tools 26 (02) (2017) 1760011.

[36] M. Kejriwal, P. Szekely, Supervised typing of big graphs using semantic embeddings, in: Proceedings of The International Workshop on Semantic Big Data, 2017, pp. 1–6.

[37] R. Sofronova, R. Biswas, M. Alam, H. Sack, Entity typing based on rdf2vec using supervised and unsupervised methods, in: European Semantic Web Conference, Springer, 2020, pp. 203–207.

[38] T. Weller, M. Acosta, Predicting instance type assertions in knowledge graphs using stochastic neural networks, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 2111–2118.

[39] B. Kotnis, V. Nastase, Analysis of the impact of negative sampling on link prediction in knowledge graphs, arXiv preprint arXiv:1708.06816.

[40] L. Cai, W. Y. Wang, Kbgan: Adversarial learning for knowledge graph embeddings, arXiv preprint arXiv:1711.04071.

[41] P. Wang, S. Li, R. Pan, Incorporating gan for negative sampling in knowledge representation learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[42] Y. Zhang, Q. Yao, Y. Shao, L. Chen, Nscaching: simple and efficient negative sampling for knowledge graph embedding, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 614–625.

[43] Y. Zhang, Q. Yao, L. Chen, Simple and automated negative sampling for knowledge graph embedding, The VLDB Journal 30 (2) (2021) 259–285.

[44] T. Pellissier Tanon, F. Suchanek, Querying the edit history of wikidata, in: European Semantic Web Conference, Springer, 2019, pp. 161–166.

[45] L. Schmelzeisen, C. Dima, S. Staab, Wikidated 1.0: An evolving knowledge graph dataset of wikidata's revision history, arXiv preprint arXiv:2112.05003.

[46] K. AlGhamdi, M. Shi, E. Simperl, Learning to recommend items to wikidata editors, in: International Semantic Web Conference, Springer, 2021, pp. 163–181.

[47] T. Pellissier Tanon, C. Bourgaux, F. Suchanek, Learning how to correct a knowledge base from the edit history, in: The World Wide Web Conference, 2019, pp. 1465–1475.

[48] T. Pellissier Tanon, F. Suchanek, Neural knowledge base repairs, in: European Semantic Web Conference, Springer, 2021, pp. 287–303.

Apéndice B

Plan de Gestión de Riesgos

B.1. Metodología

B.1.1. Metodología general

La metodología a emplear se basa en la definida en el PMBOK para la gestión de riesgos de un proyecto. Por lo tanto, se llevarán a cabo los siguientes procesos:

1. **Gestión del plan de riesgos**, donde se define cómo se van a llevar a cabo las actividades de gestión de riesgos dentro del proyecto. En este proceso se incluye la definición y actualización del plan de riesgos actual.

2. **Identificación de riesgos del proyecto**, junto con las fuentes generales de riesgos y sus características generales.

3. **Análisis cualitativo de los riesgos**, priorizándolos a partir de unos valores de probabilidad e impacto asignados a cada riesgo junto con una matriz probabilidad-impacto definida para el proyecto.

4. **Análisis cuantitativo de los riesgos**, en el que se analizará el efecto de cada riesgo en el proyecto. Este proceso incluirá el cálculo de posibles costes en el proyecto de cada riesgo.

5. **Planificar las respuestas a los riesgos**. Se establecerán indicadores que nos permitan tener un control de cada riesgo, y estrategias que llevar a cabo en función de los resultados que nos devuelva cada indicador.

6. **Implementar las respuestas a los riesgos**, basándonos en los distintos planes y respuestas acordadas previamente.

7. Por último, se llevará a cabo una **monitorización de los riesgos** durante la ejecución del proyecto. Se llevarán a cabo análisis periódicos donde se podrá actualizar el estado de los riesgos detectados y añadir o eliminar riesgos nuevos a partir de la evolución del proyecto.

B.1.2. Herramientas y técnicas

A lo largo de los distintos procesos de gestión de riesgos se utilizarán las distintas herramientas y técnicas:

- Tormenta de ideas: El alumno realizará a cabo una tormenta de ideas para distintas fases de la gestión de riesgos, como por ejemplo para el proceso de identificación de riesgos.

- Juicio de expertos: En diversos procesos de la gestión de riesgos (incluyendo, por ejemplo, la creación de este mismo plan) se reservará un tiempo de las reuniones con el director de proyecto para obtener una retroalimentación sobre la forma en la que se están gestionando los riesgos del proyecto.
- Listas de verificación: Se utilizarán listas de verificación para ayudar en el proceso de identificación de riesgos. Estas listas contienen algunos de los riesgos más comunes y generales que se han ido encontrando en los proyectos llevados a cabo por el alumno. Debido a la falta de experiencia en nuestro caso, la lista de verificación contiene riesgos detectados durante el Trabajo Fin de Grado.

B.2. Roles y responsabilidades

Se han detectado los siguientes roles principales que estarán involucrados en la gestión de riesgos:

- Director del proyecto: Posee las funciones de supervisar la gestión de los riesgos. Se encarga de igual modo de la resolución de conflictos surgidos del manejo de estos.
- Responsable de riesgos: Cada riesgo tendrá como asignado un responsable, y como tal se encargará de coordinar las acciones que se deberán llevar a cabo para minimizar el impacto de los riesgos en cuestión.
- Miembros del comité de gestión de riesgos: Los miembros del comité de gestión de riesgos deberán reunirse tanto durante los procesos de planificación como luego periódicamente para monitorizar y actualizar los riesgos del proyecto durante su ejecución.

Dado que el proyecto cuenta de tan sólo dos integrantes (el director de proyecto y el alumno), el alumno hará el rol de responsable de cada riesgo y de miembro del comité de gestión de riesgos durante el proyecto.

B.3. Categorías de riesgo

A fin de poder identificar los riesgos y conocer la estructura de los mismos se categorizan dentro de una de las siguientes categorías, pudiendo pertenecer cada uno de ellos a más de una categoría:

1. Técnico
    a) Tecnología
    b) Calidad
    c) Resultados de investigación
2. Organizacional
    a) Dependencias del proyecto
    b) Recursos
    c) Financiación
    d) Personal
3. Gestión del proyecto
    a) Estimación

Cuadro B.1. Definiciones de probabilidad

Nombre	Porcentaje	Descripción
Muy baja	0 % a 10 %	La probabilidad de que el riesgo se produzca es muy baja
Baja	10 % a 30 %	La probabilidad de que el riesgo se produzca es algo baja
Media	30 % a 55 %	Las probabilidades de que el riesgo se produzca son considerables
Alta	55 % a 75 %	Hay altas probabilidades de que la situación que describe el riesgo se de en el contexto del proyecto
Muy alta	75 % a 100 %	Hay muchas opciones de que el riesgo se produzca

   b) Planificación
   c) Control
   d) Comunicación

4. Externo
   a) Proveedores
   b) Usuario
   c) Investigaciones externas
   d) Revisores externos
   e) Otros fenómenos externos

Las categorías se crearon a partir de la estructura de desglose de riesgo definida en el PMBOK, realizando pequeñas adaptaciones para enfocarla más al ámbito del proyecto a realizar. Por lo general se han añadido categorías correspondientes a proyectos de investigación que se suelen realizar en el ámbito de una empresa.

B.4. Definiciones de probabilidad

Las definiciones de probabilidad creadas para este plan de riesgo se muestran en la tabla B.1.

B.5. Definiciones de impacto por objetivos

En la Tabla B.2 se muestran las definiciones de impacto para cada uno de los objetivos definidos en el proyecto. Para obtener el impacto total de un riesgo se calcula la media de los valores de impacto asignados a cada una de las categorías. De esta forma buscamos tener en cuenta aquellos riesgos que afectan a varios objetivos del proyecto simultáneamente. Dado que el cálculo de media hace que se obtengan unas puntuaciones menores que utilizar el valor máximo de los impactos, tanto la tolerancia al riesgo como la matriz probabilidad e impacto se han ajustado teniendo en cuenta este cálculo.

Cuadro B.2. Definiciones de impacto por objetivos

Impacto sobre los objetivos principales					
Objetivos	Escalas relativas				
	Muy bajo	Bajo	Moderado	Alto	Crítico
Coste	Incremento en el coste menor al 3 %	Incremento en el coste entre el 3 % y el 7 %	Incremento en el coste entre el 7 % y el 15 %	Incremento en el coste entre el 15 % y el 30 %	Incremento en el coste superior al 30 %
Tiempo	Incremento en el tiempo de menos de 16 horas	Incremento de menos de un 5 % del tiempo planificado	Incremento entre un 5 % y un 10 % del tiempo planificado	Incremento entre un 10 % y un 25 % del tiempo planificado	Incremento mayor a un 25 % del tiempo planificado
Alcance	El alcance sufre unas reducciones poco apreciables (se siguen cumpliendo todos los requisitos definidos)	Se reduce el alcance de algunos módulos o partes de un entregable	Se incumplen varios de los objetivos definidos para un entregable, pero este puede seguir siendo entregado	Alguno de los entregables definidos para el proyecto no puede ser entregado	Varios entregables definidos para el proyecto no pueden ser entregados
Calidad	Se reduce la calidad de algunos módulos de forma inapreciable	Se reduce la calidad de algún módulo de forma leve	Reducción de la calidad en el prototipo de validación o benchmark creados, sin implicar una cancelación del proyecto	Reducción en la calidad que implique no enviar el artículo a la revista de investigación	Reducción en la calidad que haga que alguno de los elementos entregados (validador, benchmarks...) no pueda ser usado
Resultados de investigación	Hasta el 20 % de los resultados de una pregunta de investigación se verían afectados	Entre un 20 % y un 40 % de los resultados de una pregunta de investigación se verían afectados	O bien se ve afectada una pregunta de investigación casi al completo o varias preguntas de investigación se ven algo afectadas	Los resultados de varias preguntas de investigación se ven afectados (entre 40 % y 85 %)	Los resultados de varias preguntas de investigación se ven gravemente afectados (+85 %)

Cuadro B.3. Matriz de probabilidad-impacto utilizada

			Impacto				
			Inapreciable	Bajo	Medio	Alto	Crítico
			0,07	0,15	0,30	0,63	0,95
	Muy Alta	0,90	0,06	0,14	0,27	0,57	0,86
	Alta	0,70	0,05	0,11	0,21	0,44	0,67
Prob.	Media	0,50	0,04	0,08	0,15	0,32	0,48
	Baja	0,30	0,02	0,05	0,09	0,19	0,29
	Muy Baja	0,10	0,01	0,02	0,03	0,06	0,10

B.6. Matriz de probabilidad e impacto

La matriz de probabilidad impacto utilizada se ha ajustado para tener en cuenta que el cálculo de la puntuación final se realiza a partir de la media de los impactos y no del máximo. Debido a eso, se aumentaron los valores de cada categoría de impacto. Para la probabilidad se sigue una escala lineal a la hora de asignar valores, mientras que para el impacto se sigue una escala exponencial. La matriz resultante se muestra en la Tabla B.3.

B.7. Tolerancias de los interesados

El umbral de riesgo de los interesados se ha establecido en el 0.30. Los riesgos que superen este umbral serán considerados como una verdadera amenaza para el proyecto, y deberán ser priorizados y gestionados con mayor precaución.

B.8. Formatos de los informes

Para los riesgos priorizados que requieran de un análisis complementario, el análisis se realizará sobre documentos de análisis de riesgos separados. Cada documento de análisis de riesgo contendrá la siguiente información:

- Factores del riesgo.
- Indicadores del riesgo.
- Modo de evaluar los indicadores detectados.
- Descripción de la relación de los factores de riesgo y los indicadores, así como de los planes de monitorización.
- Plan de contingencia definido.
- Riesgos derivados.
- Riesgo residual tras aplicar las respuestas establecidas.
- Presupuesto para contingencias.
- Planificación temporal de las contingencias.

A medida que surjan nuevos riesgos o se actualice el impacto de los riesgos existentes podrán crearse nuevos documentos de análisis de riesgo.

Apéndice C

Hojas de riesgos

En las siguientes páginas se adjuntan las hojas de riesgo desarrolladas para cada uno de los riesgos priorizados.

Riesgo RI-2 Hoja de Datos del Riesgo

Título del Proyecto: Validación de grafos de conocimiento utilizando el historial de ediciones

ID: 2	Nombre: Los recursos computacionales necesarios para llevar a cabo la investigación son superiores a los planificados

Descripción: Durante la fase de planificación se han estimado los recursos computacionales necesarios para llevar a cabo la investigación. Sin embargo, hay probabilidades de que estos recursos no sean los suficientes ya que no es posible saber de antemano la cantidad de datos que se obtendrá de grafos de conocimiento ni los recursos computacionales que requerirá cada método de validación probado.

Categoría(s) de riesgo:

Recursos (organizacional)

Estimación (gestión del proyecto)

Probabilidad	Impacto				Impacto Total	Respuestas
	Presupuesto	Planificación	Alcance	Calidad		
Alta	Alto	Medio	Alto	Medio	**0.385**	Se establecerán una serie de indicadores que medirán la cantidad de datos con los que estamos trabajando y los recursos tanto de RAM como de tiempo de entrenamiento y ejecución de los validadores con estos datos. Si los resultados del indicador nº4 indican que se requiere una RAM superior a la que tenemos disponible o se estima un tiempo de validación un 50% superior al planificado, se hará uso de máquinas en la nube de terceros para poder ejecutar los métodos de validación y obtener sus resultados.

Riesgo RI-2 Hoja de Datos del Riesgo

Probabilidad final	Impacto final				Impacto total final	Resultado final:
	Presupuesto	Planificación	Alcance	Calidad		El riesgo se acabó produciendo, requiriendo la necesidad de utilizar máquinas en la nube para poder entrenar algunos modelos. Con el presupuesto para contingencias de este riesgo se pudieron cubrir todos los gastos.
Muy alta	Alto	Alto	Bajo	Bajo	**0.495**	

Riesgos derivados de éste:

- Retrasos en la obtención de recursos para la máquina en la nube. En ocasiones, y dependiendo del proveedor, es necesario solicitar permisos para poder crear máquinas con ciertos recursos (especialmente si son recursos costosos). Un retraso en la concesión de estos recursos repercutiría en cambios en la planificación de las tareas relacionadas.

Riesgo residual:

Aunque se estudie el uso de cloud computing para aquellas tareas que no puedan ser manejadas con los recursos disponibles, queda un riesgo residual de que los recursos necesarios sean tan grandes que los costes asociados a la máquina sean mucho mayores al presupuesto de contingencia acordado. En ese caso, se podrían estudiar otras soluciones como trabajar con un subconjunto representativo de los datos disponibles durante la investigación.

Plan de Contingencia:

En caso de que el riesgo se produzca, se seleccionará una máquina en la nube de los distintos proveedores principales existentes (Amazon Web Services, Google Cloud y Azure) que pueda cubrir las necesidades de recursos obtenidas con los indicadores del riesgo. A partir de la máquina obtenida se instalarán las dependencias necesarias para poder ejecutar los distintos métodos de validación estudiados y se ejecutarán los métodos sobre el conjunto de datos de grafos de conocimiento colaborativos.

Presupuesto para contingencias: 2061€

Planificación temporal de las contingencias:

- **Selección de proveedor y máquina. 15-12-2021**
- **Creación de máquina y configuración de acceso. 16-12-2021**
- **Instalación de dependencias en la máquina. 16-12-2021**
- **Ejecución de métodos de validación en la máquina. 27-12-2021**

Monitorización:

11-09-21

El indicador nº1 planteado está directamente relacionado con el factor de riesgo del tamaño de los datos de investigación, el indicador 3 está relacionado con el tiempo que tarda cada método de validación y el indicador 2 con la RAM que consume cada método. El indicador nº4 sería una combinación de los 3 factores.

Los indicadores que se describen a continuación deberían ser evaluados tan sólo 1 vez en cuanto se obtenga el conjunto de datos del proyecto. Una vez se obtengan estos datos se medirán los valores de cada indicador y se estimará si el riesgo se ha producido o no a partir de éstos. El indicador 4 será el principal a manejar, ya que si obtenemos que se necesita una RAM superior a la que disponemos o el tiempo estimado de validación es un 50% superior al estimado entonces se deberían aplicar las respuestas definidas. Aún así, los indicadores previos pueden ser utilizados para ir monitorizando y actualizando las probabilidades de que el riesgo se acabe concretando o no (por ejemplo, si para el indicador 1 obtenemos que trabajamos con 50000 tripletas se podría actualizar el riesgo disminuyendo su probabilidad antes de que se obtengan los valores del resto de indicadores).

10-01-22

Se han obtenido los datos de ediciones, consistiendo en más información de la esperada. Se incrementa la probabilidad de que el riesgo ocurra en el registro, a la espera de realizar el pre-procesamiento de datos y poder evaluar los indicadores.

05-02-22

En algunos de los modelos de incrustación de grafos utilizados el indicador 4 sobrepasa el máximo de RAM de la máquina. Es necesario adquirir una máquina en la nube para entrenar alguno de los modelos. Se crean nuevas tareas para la selección y preparación de máquina, retrasando los tiempos de resolución más de lo esperado.

Indicadores:

Indicador 1: Número total de tripletas en los datos obtenidos	**Evaluación:** Una vez obtenidos los datos de grafos de conocimiento colaborativos, se calcula el número total de tripletas obtenidas sumando las tripletas de cada fuente de datos manejada.

Indicador 2: Cantidad de RAM por tripleta de los métodos de validación	**Evaluación:** Antes de entrenar y probar los distintos métodos de validación sobre los datos obtenidos, se realizará una prueba de cada uno de ellos sobre un subconjunto de datos con 100, 1000 y 10000 tripletas. Se medirá el consumo de RAM de cada método con cada conjunto de datos y a partir de ahí se realizará una estimación del consumo de RAM por tripleta de cada modelo.
Indicador 3: Cantidad de tiempo por tripleta de los métodos de validación	**Evaluación:** El proceso de evaluación de este indicador será idéntico al del indicador 2, pero en este caso para cada método se medirá el tiempo que tarda en validar cada conjunto de datos. El objetivo es estimar el tiempo que se tarda en validar cada tripleta por modelo.
Indicador 4: Estimación de recursos necesarios para ejecutar los métodos de validación sobre todos los datos obtenidos	**Evaluación:** Este indicador se obtiene a partir de una combinación de los indicadores 1, 2 y 3. Teniendo en cuenta el número total de tripletas de los datos y las estimaciones de RAM y tiempo por tripleta de cada método, se calculará la RAM y tiempo estimados para cada método de validación con el número de tripletas totales de los datos.

Riesgo RI-3 Hoja de Datos del Riesgo

Título del Proyecto: Validación de grafos de conocimiento utilizando el historial de ediciones

ID: 3	Nombre: Pérdida de datos

Descripción: La extracción de datos de revisiones es un proceso largo y del que se van a extraer una cantidad de datos considerable (estimado >1TB). Una pérdida de estos datos podría conllevar el tener que repetir el proceso de recogida y limpieza de datos por completo. Esto retrasaría el proyecto considerablemente. La pérdida puede ocurrir por múltiples motivos. Por ejemplo, un borrado por equivocación o un fallo en el disco duro de la máquina de trabajo.

Categoría(s) de riesgo:

Técnico

Probabilidad	Impacto				Impacto Total	Respuestas
	Presupuesto	Planificación	Alcance	Calidad		
Media	Alto	Crítico	Bajo	Bajo	**0.450**	Se utilizará algún sistema de almacenamiento de datos en la nube para mantener una copia de los datos recogidos para la investigación. Siempre que las licencias lo permitan, se facilitará el acceso a estos datos de forma pública a fuentes externas.

Riesgo RI-3 Hoja de Datos del Riesgo

Probabilidad final	Impacto final				Impacto total final	Resultado final:
	Presupuesto	Planificación	Alcance	Calidad		El riesgo no se acabó produciendo. Aún así se mantuvieron copias en la nube durante el desarrollo del proyecto para evitar las posibles consecuencias en caso de producirse.
Baja	Medio	Crítico	Bajo	Bajo	**0.270**	

Riesgos derivados de éste: N/A

Riesgo residual:

Aunque se mantengan copias en la nube del conjunto de datos obtenido existe el riesgo residual de que estas copias tampoco estén accesibles por problemas del proveedor enla nube. Sin embargo, la probabilidad de que este riesgo residual se produzca se considera mínima.

Plan de Contingencia:

En caso de que el riesgo se produzca, se descargará la copia de los datos almacenada en la nube y se proseguirá con el desarrollo del proyecto, incurriendo en unos retrasos prácticamente insignificantes respecto a lo planificado.

Presupuesto para contingencias: 100€

Planificación temporal de las contingencias:

N/A

Monitorización:

10-01-22

Se han obtenido los datos de ediciones. Se selecciona el proveedor Microsoft para almacenar copias de los datos en OneDrive (total -> 700GB). Debido a que se tiene un acuerdo entre la Universidad de Oviedo y Microsoft, no se incurre ningún coste adicional en el proyecto a pesar del tamaño considerable de la copia.

06-05-22

Se finaliza la experimentación del proyecto. No se ha producido el riesgo, por lo que no ha sido necesario utilizar la copia de datos.

Riesgo RI-4 Hoja de Datos del Riesgo

Título del Proyecto: Validación de grafos de conocimiento utilizando el historial de ediciones

ID: 4	Nombre: Modificaciones en el alcance del proyecto

Descripción: Debido a la naturaleza de investigación del proyecto hay más posibilidades de descubrir nueva información durante el desarrollo que abra las puertas a distintas tareas y experimentos.

Categoría(s) de riesgo:

Gestión

Probabilidad	Impacto				Impacto Total	Respuestas
	Presupuesto	Planificación	Alcance	Calidad		
Alta	Medio	Alto	Alto	Medio	**0.385**	Se establecerán reuniones periódicas (mensualmente) con el director. Una parte de estas reuniones se dedicará a tratar el estado del proyecto y posibles modificaciones a seguir. Toda modificación en los objetivos o tareas definidos en el proyecto deberá ser acordada previamente con el director.

Riesgo RI-4 Hoja de Datos del Riesgo

Probabilidad final	Impacto final				Impacto total final	Resultado final:
	Presupuesto	Planificación	Alcance	Calidad		
Alta	Bajo	Medio	Medio	Bajo	**0.275**	Fue necesario modificar algunos aspectos del análisis de historial de ediciones a partir de la revisión de literatura realizada. También fue necesario eliminar tareas de clusterización de datos y análisis de patrones en los datos de ediciones. Sin embargo, esto no supuso un impacto tan grande en el desarrollo del proyecto como el esperado.

Riesgos derivados de éste: N/A

Riesgo residual: N/A

Plan de Contingencia:

Se irán registrando periódicamente los posibles cambios al alcance del proyecto detectados durante el transcurso del mismo. En las reuniones mensuales realizadas con el director de proyecto se evaluarán estos cambios. Todo cambio deberá tener la aprobación del director de proyecto.

En caso de que el riesgo se produzca se reserva un presupuesto para posibles contingencias que tiene en cuenta las horas adicionales de trabajo en el proyecto.

Presupuesto para contingencias: 1000€

Monitorización:

01-11-21

Tras revisar las técnicas de validación de grafos de conocimiento se decide utilizar principalmente modelos de incrustación de grafos sobre otras técnicas, modificando algunas de las tareas previamente definidas. Se obtiene el visto bueno del director de proyecto para realizar este cambio.

03-02-22

Debido al volumen de datos de ediciones mayor de lo esperado, y a los retrasos actuales en el desarrollo del proyecto, se decide eliminar tareas de clusterización de datos y análisis de patrones en los datos de ediciones tras la aprobación del director de proyecto.

Riesgo RI-6 Hoja de Datos del Riesgo

Título del Proyecto: Validación de grafos de conocimiento utilizando el historial de ediciones

ID: 6	**Nombre: Planificación optimista**

Descripción: Se subestima el tiempo de las tareas del proyecto. Esto podría resultar en no cumplir los plazos esperados de finalización del proyecto, o tener que reducir la calidad y/o el alcance del proyecto para poder cumplir con los plazos.

Categoría(s) de riesgo:

Gestión

Probabilidad	**Impacto**				**Impacto Total**	**Respuestas**
	Presupuesto	**Planificación**	**Alcance**	**Calidad**		
Alta	Medio	Alto	Medio	Alto	**0.385**	El objetivo es minimizar este riesgo a través de la gestión del tiempo realizada. En primer lugar, la propia estimación de tiempos utilizando PERT tiene en cuenta escenarios optimistas y pesimistas que estabilizan más las estimaciones. En segundo lugar, se realizará un análisis de desempeño en cada hito con el fin de detectar problemas de planificación y corregirlos lo antes posible.

Riesgo RI-6 Hoja de Datos del Riesgo

Probabilidad final	Impacto final				Impacto total final	Resultado final:
	Presupuesto	Planificación	Alcance	Calidad		Se produjeron retrasos en algunas tareas del proyecto respecto a lo estimado. El presupuesto final de costes fue mayor al esperado, y la fecha final de finalización del proyecto se retrasó 10 días.
Muy Alta	Medio	Alto	Bajo	Bajo	**0.495**	

Riesgos derivados de éste: N/A

Riesgo residual: N/A

Plan de Contingencia:

Durante cada línea base se analizará el desempeño del proyecto. En las reuniones mensuales realizadas con el director de proyecto también se evaluará la situación actual respecto a la planificación.

En caso de que se detecten retrasos respecto a lo planificado se reevaluará realizar cambios en el alcance del proyecto o utilizar más días tras la fecha de finalización prevista para terminar las tareas retrasadas.

Presupuesto para contingencias: 1500€

Monitorización:

15-03-22

Se detectan retrasos en las tareas de análisis del historial de ediciones respecto a lo previsto debido al gran volumen de datos. Se decide prescindir de algunas tareas no críticas del proyecto para compensar por estos retrasos.

02-05-22

La redacción y corrección del artículo lleva considerablemente más tiempo del esperado. Se decide utilizar 10 días adicionales tras la fecha de finalización del proyecto para terminar las tareas restantes

Riesgo RI-8 Hoja de Datos del Riesgo

Título del Proyecto: Validación de grafos de conocimiento utilizando el historial de ediciones

ID: 8	**Nombre: Aparece publicación similar durante el tiempo de investigación**

Descripción: Mientras se lleva a cabo el proyecto aparece en paralelo una investigación que trata el mismo tema, pudiendo solapar parte del trabajo que ya ha sido realizado

Categoría(s) de riesgo:

Externo

Probabilidad	**Impacto**				**Impacto Total**	**Respuestas**
	Presupuesto	**Planificación**	**Alcance**	**Calidad**		
Baja	Inapreciable	Bajo	Bajo	Alto	**0.165**	Dado que el proyecto dura unos 9 meses, se añadirá una tarea en el cierre del proyecto de repasar la literatura que ha surgido durante el proyecto y actualizar las partes de la memoria afectadas con esta nueva información.
Probabilidad final	**Impacto final**				**Impacto total final**	**Resultado final:** Apareció una publicación relacionada con el tema
	Presupuesto	Planificación	Alcance	Calidad		

Media	Bajo	Medio	Bajo	Medio	**0.275**	propuesto durante el desarrollo del proyecto. En este caso, más que suponer un riesgo supuso una oportunidad, ya que ofrecía una fuente de datos adicional que podía ser aprovechada en nuestros experimentos.

Riesgos derivados de éste: N/A

Riesgo residual: N/A

Monitorización:

01-12-21

Se publica un nuevo artículo sobre obtención de datos de ediciones de Wikidata. Se añaden nuevas tareas para evaluar la incorporación de esta fuente de datos en nuestros experimentos.

28-04-22

Finaliza la tarea de revisión de nueva literatura durante el transcurso del proyecto. No se detecta ningún artículo adicional relativo al uso del historial de ediciones en grafos de conocimiento.

Apéndice D

Presupuesto Detallado

El presupuesto para el cliente que hemos mostrado en la Subsección 4.1.5 Presupuesto inicial se obtiene a través de una serie de cálculos adicionales, desglosados en este documento.

D.1. Definición del modelo de costes

Para calcular el presupuesto del proyecto empezaremos por definir los tipos de costes directos e indirectos y los distintos perfiles que tendremos para ejecutar el proyecto.

D.1.1. Perfiles

En la tabla D.1 se muestran los distintos roles que se tomarán para ejecutar el proyecto, junto con su salario bruto y coste salarial por año. El salario bruto se obtuvo utilizando como referencia el sitio web de payscale[1], seleccionando el sueldo medio en España de cada uno de los perfiles. Para el cálculo del coste salarial se añadió al salario bruto un porcentaje de costes para cubrir gastos como la Seguridad Social.

Cuadro D.1. Personal y sueldos

Perfiles	Salario bruto anual	Coste salarial anual
Jefe de investigación	47.000,00€	62.510,00€
Asistente de investigación	32.200,00€	42.826,00€
Ingeniero de ontologías	53.500,00€	71.155,00€
Jefe de proyecto	55.000,00€	73.150,00€
Programador Senior	40.000,00€	53.200,00€
Programador Junior	35.000,00€	46.550,00€
Científico de datos	46.000,00€	61.180,00€
Auxiliar de sistemas	20.000,00€	26.600,00€
Asesor financiero	37.000,00€	49.210,00€
	TOTAL:	**415.226,00€**

[1] https://www.payscale.com

D.1.2. Otros costes

Costes debidos a trabajos no productivos

De los costes salariales presentados previamente, una parte serán costes directos (tiempo directamente productivo del empleado) mientras que otra parte se corresponderá con costes indirectos (formación, temas administrativos, contratación...). En la tabla D.2 se muestran los costes directos e indirectos asociados a cada perfil. Se ha asignado manualmente un porcentaje de productividad a cada perfil, indicando cuántas horas de trabajo se corresponden con costes directos.

Cuadro D.2. Costes directos e indirectos de cada perfil

Perfiles	Productividad	Coste directo	Coste indirecto
Jefe de investigación	60 %	37.506,00€	25.004,00€
Asistente de investigación	75 %	32.119,50€	10.706,50€
Jefe de proyecto	70 %	51.205,00€	21.945,00€
Programador Senior	85 %	45.220,00€	7.980,00€
Programador Junior	77 %	35.843,50€	10.706,50€
Científico de datos	75 %	45.885,00€	15.295,00€
Auxiliar de sistemas	0 %	0€	26.600,00€
Asesor financiero	7 %	3,444.70€	45,765.30€
	TOTAL:	**251.223,70€**	**164.002,30€**

Costes de servicios

Además de los costes debidos a trabajos no productivos, también se cuenta con una serie de servicios necesarios para un correcto funcionamiento. En la tabla D.3 se muestran esos servicios y su coste tanto mensual como anual.

Cuadro D.3. Costes asociados a servicios

Servicio	Coste mes	Coste año
Limpieza	300,00€	3.600,00€
Alquiler de oficina	800,00€	9.600,00€
Coste eléctrico	400,00€	4.800,00€
Calefacción	115,00€	1.380,00€
Agua	100,00€	1.200,00€
Material de oficina	150,00€	1.800,00€
	TOTAL:	**22.380,00€**

Costes de material

Adicionalmente, también se cuenta con una serie de medios tecnológicos y herramientas necesarios para poder producir. En la tabla D.4 se muestran los costes asociados a estos medios materiales.

Cuadro D.4. Costes de material

Equipo	Precio	Coste año	Tipo	Plazo
Equipos de desarrollo	1.300,00€	325,00€	Amortiz.	4
Portátil	800,00€	200,00€	Amortiz.	4
Licencia de desarrollo PyCharm	200,00€	200,00€	Alquiler	
Servidor para la ejecución de cálculos computacionalmente costosos	3.500,00€	583,33€	Amortiz.	6

Costes indirectos adicionales

Por último, también hay una serie de costes indirectos adicionales que no entran en ninguna de las categorías presentadas previamente. En la tabla D.5 se muestran estos costes.

Cuadro D.5. Costes de material

Concepto	Coste
Formación	7.500,00€
Costes financieros	11.070,65€
Imprevistos	10.000,00€
TOTAL:	**28.570,65€**

Cálculo de los costes indirectos

Si sumamos todos los costes indirectos presentados previamente, el total de costes indirectos es de **194.202,87€** y el de costes directos de **251.223,70€**. Esto hace que se tenga un total de costes anuales de **445.426,57€**. Suponiendo un beneficio esperado del **20 %** (89.085,31€), se tendría una necesidad de facturación de **545.511,88€**.

D.1.3. Precios hora

Para poder cubrir las necesidades de facturación presentadas previamente, pasamos a definir los precios por hora de cada uno de los perfiles definidos. En la tabla D.6 se muestran estos cálculos. En primer lugar, se definen las horas productivas para cada perfil a partir del porcentaje de productividad que se les ha asignado previamente. Posteriormente se les asigna un precio hora que cubra los costes anuales (precio hora sin beneficios) y otro precio hora que además incluye los beneficios (precio hora cliente). Estos últimos se usarán sólo en los presupuestos por horas para el cliente.

Cuadro D.6. Precio por hora de cada perfil

Perfiles	Precios hora (sólo costes directos)	Precio hora cliente	Precio hora (sin beneficios)
Jefe de investigación	31,13€	66,23€	55,20€
Asistente de investigación	21,33€	45,38€	37,81€
Jefe de proyecto	36,43€	77,51€	64,59€
Programador Senior	26,49€	56,37€	46,97€
Programador Junior	23,18€	49,32€	41,10€
Científico de datos	30,47€	64,82€	54,02€
Auxiliar de sistemas	-	-	-
Asesor financiero	24,51€	52,14€	43,45€

D.1.4. Resumen

A modo de resumen se muestra la tabla D.7.

Cuadro D.7. Resumen del modelo de costes

Concepto	Importe
Total de los costes directos	251.223,70€
Total de los costes indirectos	194.202,87€
Total de costes directos + indirectos	445.426,57€
Beneficio deseado (20 %)	89.085,31€

D.2. Presupuesto inicial

D.2.1. Presupuesto interno

Partidas del proyecto

El proyecto propuesto está formado por las siguientes partidas:

- Partida 1: Estudio de validación de grafos de conocimiento con el historial de ediciones.
- Partida 2: Desarrollo de prototipo de validación.
- Partida 3: Divulgación de resultados.
- Partida 4: Gestión del proyecto.
- Partida 5: Otros costes.

Las 3 primeras partidas serán presentadas directamente al cliente, ya que generan productos con valor directo para él. Sin embargo, las 2 últimas partidas sólo se calcularán a nivel interno, y sus costes se distribuirán entre el resto de partidas en el presupuesto final del cliente[2].

Elaboración del presupuesto interno

A continuación pasamos a mostrar los conceptos presentes en cada partida, los recursos y unidades utilizadas, así como el coste total. Para todas las partidas, las horas de recursos de cada concepto se obtienen a partir de la planificación presentada en la sección 4.1. En caso de otro tipo de recursos (p.ej. dietas), se mencionará explícitamente en la partida correspondiente el origen de la cantidad del recurso.

Partida 1: Estudio de validación de grafos de conocimiento con el historial de ediciones. En esta partida se incluyen los conceptos relacionados con el estudio del uso del historial de ediciones para la validación de grafos de conocimiento. En la tabla D.8 se muestran estos conceptos, y los recursos utilizados.

Cuadro D.8. Presupuesto de costes de la partida 1

Partida 1: Estudio de validación de grafos de conocimiento utilizando el historial de ediciones									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
1			**Revisión de literatura y estado del arte**						7.131,22€
	1		Uso del historial de revisiones en grafos de conocimiento					1.346,76€	
		1	Jefe de investigación	24,4	horas	55,20€	1.346,76€		
	2		Algoritmos de corrección de grafos de conocimiento					2.649,37€	

[2]Aunque las partidas 4 y 5 sean necesarias para completar el proyecto, no aportan ningún valor al cliente y por lo tanto no se mostrarán en su presupuesto.

Cuadro D.8. Presupuesto de costes de la partida 1

Partida 1: Estudio de validación de grafos de conocimiento utilizando el historial de ediciones									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
		1	Jefe de investigación	28	horas	55,20€	2.649,37€		
	3		Dinámicas de conflicto y colaboración en Wikipedia y Wikidata					1.788,32€	
		1	Jefe de investigación	32,4	horas	55,20€	1.788,32€		
	4		Métodos de análisis de grafos					1.356,76€	
		1	Jefe de investigación	24,4	horas	55,20€	1.346,76€		
2			**Desarrollo de la investigación**						21.811,91€
	1		Obtención de datos					5.747,82€	
		1	Asistente de investigación	152	horas	37,81€	5.747,82€		
	2		Identificación de patrones de edición					5.097,41€	
		1	Asistente de investigación	134,8	horas	37,81€	5.097,41€		
	3		Uso de historial de ediciones en validación de grafos					10.966,69€	
		1	Jefe de investigación	157,2	horas	55,20€	8.676,69€		
		2	Reserva de contingencia	1		2.290,00€	2.290,00€		
3			**Publicación de código de investigación**						2.117,71€
	1		Retoques finales en el código implementado					332,77€	
		1	Asistente de investigación	8,8	horas	37,81€	332,77€		
	2		Ejecución de todo el código de investigación (reproducibilidad)					712,38€	
		1	Asistente de investigación	17,6	horas	37,81€	712,38€		
	3		Documentación de pasos para reproducir los resultados					323,81€	

Cuadro D.8. Presupuesto de costes de la partida 1

Partida 1: Estudio de validación de grafos de conocimiento utilizando el historial de ediciones									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
		1	Asistente de investigación	8	horas	37,81€	323,81€		
4			**Análisis final de resultados de investigación**						874,39€
	1		Respuesta a las preguntas de investigación planteadas					331,17€	
		1	Jefe de investigación	6	horas	55,20€	331,17€		
	2		Definición de áreas de mejora y trabajo futuro					485,72€	
		1	Jefe de investigación	8,8	horas	55,20€	485,72€		

TOTAL:	31.877,73€

Partida 2: Desarrollo de prototipo de validación. En la partida 2 se incluyen los conceptos relacionados con el desarrollo de un prototipo software que permita validar grafos de conocimiento a partir de los resultados experimentales del proyecto. En la tabla D.9 se muestra el presupuesto de la partida.

Cuadro D.9. Presupuesto de costes de la partida 2

Partida 2: Desarrollo de prototipo de validación									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
1			**Implementación del prototipo**						657,64€
	1		Carga de modelos creados					244,27€	
		1	Programador Senior	5,2	horas	46,97€	244,27€		
	2		Implementación funcionalidad de validación					413,38€	
		1	Programador Senior	8,8	horas	46,97€	413,38€		
2			**Documentación de instalación y uso**						197,29€
	1		Documentación de instalación y uso					197,29€	

Cuadro D.9. Presupuesto de costes de la partida 2

Partida 2: Desarrollo de prototipo de validación									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
		1	Programador Senior	4,2	horas	46,97€	197,29€		
3			**Pruebas**						826,75€
	1		Pruebas de carga y rendimiento					394,59€	
		1	Programador Senior	8,4	horas	46,97€	394,59€		
	2		Pruebas metamórficas del validador					432,17€	
		1	Programador Senior	9,2	horas	46,97€	432,17€		

TOTAL:	1.681,69€

Partida 3: Divulgación de resultados. Esta partida se corresponde con la creación de materiales de divulgación de la investigación realizada (publicación de artículo en una revista de impacto, diapositivas de presentación...). En la tabla D.10 se muestra el presupuesto de la partida.

Cuadro D.10. Presupuesto de costes de la partida 3

Partida 3: Divulgación de resultados									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
1			**Publicación de artículo**						2.747,85€
	1		Selección de revista					332,77€	
		1	Asistente de investigación	8,8	horas	37,81€	332,77€		
	2		Escribir artículo					1.573,09€	
		1	Asistente de investigación	41,86	horas	37,81€	1.573,09€		
	3		Revisión interna del artículo					463,64€	
		1	Jefe de investigación	8,4	horas	55,20€	463,64€		
	4		Corrección de revisión interna					317,64€	
		1	Asistente de investigación	8,4	horas	37,81€	317,64€		
	3		Envío de artículo a revista					60,71€	
		1	Jefe de investigación	1,1	horas	55,20€	60,71€		

Cuadro D.10. Presupuesto de costes de la partida 3

Partida 3: Divulgación de resultados									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
2			**Creación de página web con resultados del proyecto**						1.052,23€
	1		Creación de página web con resultados del proyecto					1.052,23€	
		1	Programador Junior	25,6	horas	41,10€	1.052,23€		
3			**Demostración online interactiva**						826,75€
	1		Demostración online interactiva					826,75€	
		1	Programador Senior	17,6	horas	46,97€	826,75€		
4			**Creación de diapositivas sobre la investigación**						877,30€
	1		Creación de diapositivas sobre la investigación					877,30€	
		1	Asistente de investigación	23,2	horas	37,81€	877,30€		
5			**Reserva de contingencia para riesgo RI5**						1.200,00€
	1		Reserva de contingencia					1.200,00€	
		1	Reserva de contingencia	1		1.200,00€	1.200,00€		

TOTAL:	6.704,14€

Partida 4: Gestión del proyecto En esta partida se incluyen las tareas de gestión del proyecto propuesto. En la tabla D.11 se muestra el presupuesto de la partida.

Cuadro D.11. Presupuesto de costes de la partida 4

Partida 4: Gestión del proyecto									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
1			**Planificación del proyecto**						5.226,68€

Cuadro D.11. Presupuesto de costes de la partida 4

Partida 4: Gestión del proyecto									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
	1		Definición de objetivos y alcance del proyecto					568,39€	
		1	Jefe de proyecto	8,8	horas	64,59€	568,39€		
	2		Creación de acta de proyecto					568,39€	
		1	Jefe de proyecto	8,8	horas	64,59€	568,39€		
	3		Desglose en tareas					800,92€	
		1	Jefe de proyecto	12,4	horas	64,59€	800,92€		
	4		Estimar duraciones de las tareas					671,74€	
		1	Jefe de proyecto	10,4	horas	64,59€	671,74€		
	5		Estructura inicial de la memoria					452,60€	
		1	Jefe de investigación	8,2	horas	55,20€	452,60€		
	6		Definición de lista de riesgos					788,00€	
		1	Jefe de proyecto	12,2	horas	64,59€	788,00€		
	7		Estudio de medidas a adoptar					284,20€	
		1	Jefe de proyecto	4,4	horas	64,59€	284,20€		
	8		Refinar tareas con las medidas adoptadas					142,10€	
		1	Jefe de proyecto	2,2	horas	64,59€	142,10€		
	9		Cálculo de costes del proyecto					280,16€	
		1	Jefe de proyecto	2,4	horas	64,59€	155,02€		
		2	Asesor financiero	2,4	horas	52,14€	125,14€		
	10		Creación de presupuesto interno					413,38€	
		1	Jefe de proyecto	6,4	horas	64,59€	413,38€		
	11		Creación de presupuesto cliente					256,81€	
		1	Jefe de proyecto	2,2	horas	64,59€	142,10€		
		2	Asesor financiero	2,2	horas	52,14€	114,71€		
2			**Monitorización**						3.898,61€
	1		Monitorización del estado de la planificación del proyecto					930,10€	
		1	Jefe de proyecto	14,4	horas	64,59€	930,10€		

Cuadro D.11. Presupuesto de costes de la partida 4

Partida 4: Gestión del proyecto									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
	2		Monitorización y actualización de riesgos del proyecto					1.291,80€	
		1	Jefe de proyecto	20	horas	64,59€	1.291,80€		
	3		Monitorización y actualización del presupuesto del proyecto					630,35€	
		1	Jefe de proyecto	5,4	horas	64,59€	348,79€		
		2	Asesor financiero	5,4	horas	52,14€	281,57€		
	4		Mantenimiento y actualización de bitácora del proyecto					1.046,36€	
		1	Jefe de proyecto	16,2	horas	64,59€	1.046,36€		
3			**Cierre del proyecto**						2.542,90€
	1		Creación de planificación final					439,21€	
		1	Jefe de proyecto	6,8	horas	64,59€	439,21€		
	2		Creación de presupuesto final					536,97€	
		1	Jefe de proyecto	4,6	horas	64,59€	297,11€		
		2	Asesor financiero	4,6	horas	52,14€	239,85€		
	3		Análisis de planificación final					213,15€	
		1	Jefe de proyecto	3,3	horas	64,59€	213,15€		
	4		Análisis de presupuesto final					284,20€	
		1	Jefe de proyecto	4,4	horas	64,59€	284,20€		
	5		Documentación de aspectos positivos y negativos del desempeño en el proyecto					142,10€	
		1	Jefe de proyecto	2,2	horas	64,59€	142,10€		
	6		Ajustes a la memoria antes de la entrega					927,28€	
		1	Jefe de investigación	16,8	horas	55,20€	927,28€		

TOTAL:	11.668,19€

Cuadro D.13. Resumen del presupuesto de costes interno

Cod.	Partida	Total
01	Estudio de validación de grafos de conocimiento utilizando el historial de ediciones	31.877,73€
02	Desarrollo de prototipo de validación	1.682,69€
03	Divulgación de resultados	6.704,14€
04	Gestión del proyecto	11.668,19€
05	Otros costes	3.016,72€
	TOTAL:	**54.948,47€**

Partida 5: Otros costes. En esta partida se incluyen otros costes del proyecto que no pertenecen a ninguna de las partidas anteriores (viajes, reuniones con el cliente...). En la tabla D.12 se muestra el presupuesto de esta partida.

Cuadro D.12. Presupuesto de costes de la partida 5

Partida 5: Otros costes									
I1	**I2**	**I3**	**Descripción**	**Cantidad**	**Uds.**	**Precio**	**Subtotal (3)**	**Subtotal (2)**	**Total**
1			**Otros costes**						3.016,72€
	1		Reservas					2.500,00€	
		1	Imprevistos y reservas	1		2.500,00€	2.500,00€		
	2		Reuniones con el directos					516,72€	
		1	Jefe de proyecto	8	horas	64,59€	516,72€		

TOTAL:	3.016,72€

Resumen

En la tabla D.13 se muestra el presupuesto de costes interno compuesto de todas las partidas presentadas previamente. El total de costes internos del proyecto asciende a **54.948,47€**. Cabe destacar que estos costes no incluyen ningún tipo de beneficio. En el siguiente capítulo se añadirá este beneficio y se mostrarán las partidas que se presentarán al cliente.

D.2.2. Presupuesto de cliente

Elaboración

Para la elaboración del presupuesto del cliente es importante primero saber qué información se mostrará en el presupuesto. Como se ha mencionado previamente, las partidas 4 y 5 no se mostrarán al cliente, por lo que será necesario repartir los costes de estas partidas entre el

Cuadro D.14. Presupuesto de costes de las partidas del cliente (sin beneficios)

Partidas	Conceptos	Precio
Partida 1	Obtención de datos	8.652,65€
	Identificación de patrones de edición	8.002,24€
	Uso de historial de ediciones en validación de grafos	14.871,52€
	Código fuente de la investigación	5.022,54€
Partida 2	Implementación de prototipo de validador	5.352,92€
Partida 3	Artículo de investigación	5.622,12€
	Materiales complementarios	4.753,25€
	TOTAL:	**51.277,24€**

resto. Además, algunos de los conceptos de las otras partidas tampoco se mostrarán[3] y su coste deberá repartirse entre el resto.

Teniendo esto en cuenta, en la tabla D.14 se muestra el presupuesto de costes de las partidas del cliente. Este presupuesto incluye todas las partidas que se mostrarán al cliente, en las que se han distribuido el coste del resto de partidas internas con el fin de cubrir los costes totales del proyecto. Lo único que quedaría ahora para obtener el presupuesto final del cliente es aplicar los beneficios del proyecto. Partiendo de la suposición de que se espere obtener un 20 % de los beneficios, se repartirán 10.255,45€ entre todas las partidas del presupuesto.

Presupuesto de cliente desarrollado

Con el beneficio aplicado a las partidas mostradas previamente, se obtiene el presupuesto del cliente. En la tabla D.15 se muestra el presupuesto completo del cliente, mientras que en la tabla D.16 se muestra el presupuesto resumido (sin desarrollar los conceptos de las partidas).

D.3. Presupuesto final

En la Tabla D.17 se muestra el presupuesto interno final (sin beneficios) teniendo en cuenta los cambios en la planificación mostrados. Podemos ver como el coste total del proyecto fue 8.319,01€ superior al esperado. Aún así, teniendo en cuenta el margen de beneficios del 20 % especificado para el presupuesto del cliente, el proyecto seguiría teniendo beneficios (sólo que en vez de 11.919€ se tendrían tan sólo 3.600€ de beneficios).

Los datos detallados del presupuesto, con los conceptos específicos que sufrieron modificaciones de costes y un nuevo hipotético precio al cliente en el que mantendríamos el 20 % de beneficios esperados se muestran en el archivo '*presupuesto_final.xlsx*' adjunto a esta entrega.

[3]Por ejemplo, el concepto de revisión de literatura dentro de la partida 1. En general, se mostrarán aquellos conceptos que aporten un valor directo al cliente.

Cuadro D.15. Presupuesto de cliente

Cod.	Item	Partida	Importe	Total
1		Estudio de validación de grafos de conocimiento utilizando el historial de ediciones		42.658,75€
	1	Obtención de datos	10.383,18€	
	2	Identificación de patrones de edición	9.602,69€	
	3	Uso de historial de ediciones en validación de grafos	16.645,82€	
	4	Código fuente de la investigación	6.027,05€	
2		Implementación de prototipo de validador		6.424,51€
3		Divulgación de resultados		12.450,44€
	1	Artículo de investigación	6.746,54€	
	2	Otros materiales complementarios	5.703,90€	
			Subtotal	**61.532,69€**
			IVA (21 %)	**12.921,86€**
			TOTAL CLIENTE:	**74.454,55€**

Cuadro D.16. Presupuesto de cliente (resumido)

Cod.	Partida	Total
1	Estudio de validación de grafos de conocimiento utilizando el historial de ediciones	42.658,75€
2	Implementación de prototipo de validador	6.424,51€
3	Divulgación de resultados	12.450,44€
	Subtotal	**61.532,69€**
	IVA (21 %)	**12.921,86€**
	TOTAL CLIENTE:	**74.454,55€**

Cuadro D.17. Presupuesto final de costes de las partidas del cliente (sin beneficios)

Partidas	**Conceptos**	**Precio**
Partida 1	Obtención de datos	11.656,20€ (+3.003,55€)
	Identificación de patrones de edición	7.799,12€ (-203,12€)
	Uso de historial de ediciones en validación de grafos	15.016,29€ (+144,77)
	Código fuente de la investigación	5.681,59€ (+659,05€)
Partida 2	Implementación de prototipo de validador	5.822,56€ (+469,64€)
Partida 3	Artículo de investigación	7.958,49€ (+2336.37€)
	Materiales complementarios	5.662,05€ (+908,80€)
	TOTAL:	**59.596,29€ (+8.319,01€)**

Referencias

[1] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. d. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, *et al.*, "Knowledge graphs," *Synthesis Lectures on Data, Semantics, and Knowledge*, vol. 12, no. 2, pp. 1–257, 2021.

[2] "Introducing the knowledge graph: things, not strings." http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html. Accessed: 2022-05-30.

[3] F. Cifuentes-Silva and J. E. Labra Gayo, "Legislative document content extraction based on semantic web technologies," in *European Semantic Web Conference*, pp. 558–573, Springer, 2019.

[4] D. N. Slenter, M. Kutmon, K. Hanspers, A. Riutta, J. Windsor, N. Nunes, J. Mélius, E. Cirillo, S. L. Coort, D. Digles, *et al.*, "Wikipathways: a multifaceted pathway database bridging metabolomics to other omics research," *Nucleic acids research*, vol. 46, no. D1, pp. D661–D667, 2018.

[5] H. García-González, J. E. L. Gayo, and M. Paule-Ruiz, "Enhancing e-learning content by using semantic web technologies," *IEEE Transactions on Learning Technologies*, vol. 10, no. 4, pp. 544–550, 2016.

[6] A. Nayak, V. Kesri, and R. K. Dubey, "Knowledge graph based automated generation of test cases in software engineering," in *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, pp. 289–295, 2020.

[7] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic web*, vol. 6, no. 2, pp. 167–195, 2015.

[8] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[9] A. Piscopo and E. Simperl, "What we talk about when we talk about wikidata quality: a literature survey," in *Proceedings of the 15th International Symposium on Open Collaboration*, pp. 1–11, 2019.

[10] M. Färber, F. Bartscherer, C. Menne, and A. Rettinger, "Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago," *Semantic Web*, vol. 9, no. 1, pp. 77–129, 2018.

[11] K. Shenoy, F. Ilievski, D. Garijo, D. Schwabe, and P. Szekely, "A study of the quality of wikidata," *Journal of Web Semantics*, vol. 72, p. 100679, 2022.

[12] P. F. Patel-Schneider, "Using description logics for rdf constraint checking and closed-world recognition," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[13] E. Prud'hommeaux, J. E. Labra Gayo, and H. Solbrig, "Shape expressions: an rdf validation and transformation language," in *Proceedings of the 10th International Conference on Semantic Systems*, pp. 32–40, 2014.

[14] H. Paulheim and C. Bizer, "Improving the quality of linked data using statistical distributions," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 10, no. 2, pp. 63–86, 2014.

[15] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," *Advances in neural information processing systems*, vol. 26, 2013.

[16] J. Sleeman and T. Finin, "Type prediction for efficient coreference resolution in heterogeneous semantic graphs," in *2013 IEEE Seventh International Conference on Semantic Computing*, pp. 78–85, IEEE, 2013.

[17] P. Ristoski and H. Paulheim, "Rdf2vec: Rdf graph embeddings for data mining," in *International Semantic Web Conference*, pp. 498–514, Springer, 2016.

[18] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, (New Orleans, Louisiana), pp. 327–333, Association for Computational Linguistics, June 2018.

[19] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[20] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, 2014.

[21] T. Steiner, "Bots vs. wikipedians, anons vs. logged-ins (redux) a global study of edit activity on wikipedia and wikidata," in *Proceedings of The International Symposium on Open Collaboration*, pp. 1–7, 2014.

[22] C. Müller-Birn, B. Karran, J. Lehmann, and M. Luczak-Rösch, "Peer-production system or collaborative ontology engineering effort: What is wikidata?," in *Proceedings of the 11th International Symposium on Open Collaboration*, pp. 1–10, 2015.

[23] A. Piscopo, C. Phethean, and E. Simperl, "Wikidatians are born: Paths to full participation in a collaborative structured knowledge base," 2017.

[24] C. Sarasua, A. Checco, G. Demartini, D. Difallah, M. Feldman, and L. Pintscher, "The evolution of power and standard wikidata editors: comparing editing behavior over time to predict lifespan and volume of edits," *Computer Supported Cooperative Work (CSCW)*, vol. 28, no. 5, pp. 843–882, 2019.

[25] A. Piscopo, C. Phethean, and E. Simperl, "What makes a good collaborative knowledge graph: group composition and quality in wikidata," in *International Conference on Social Informatics*, pp. 305–322, Springer, 2017.

[26] A. Piscopo and E. Simperl, "Who models the world? collaborative ontology creation and user roles in wikidata," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–18, 2018.

[27] A. Melo, J. Völker, and H. Paulheim, "Type prediction in noisy rdf knowledge bases using hierarchical multilabel classification with graph and latent features," *International Journal on Artificial Intelligence Tools*, vol. 26, no. 02, p. 1760011, 2017.

[28] M. Kejriwal and P. Szekely, "Supervised typing of big graphs using semantic embeddings," in *Proceedings of The International Workshop on Semantic Big Data*, pp. 1–6, 2017.

[29] R. Sofronova, R. Biswas, M. Alam, and H. Sack, "Entity typing based on rdf2vec using supervised and unsupervised methods," in *European Semantic Web Conference*, pp. 203–207, Springer, 2020.

[30] T. Weller and M. Acosta, "Predicting instance type assertions in knowledge graphs using stochastic neural networks," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2111–2118, 2021.

[31] B. Kotnis and V. Nastase, "Analysis of the impact of negative sampling on link prediction in knowledge graphs," *arXiv preprint arXiv:1708.06816*, 2017.

[32] L. Cai and W. Y. Wang, "Kbgan: Adversarial learning for knowledge graph embeddings," *arXiv preprint arXiv:1711.04071*, 2017.

[33] P. Wang, S. Li, and R. Pan, "Incorporating gan for negative sampling in knowledge representation learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[34] Y. Zhang, Q. Yao, Y. Shao, and L. Chen, "Nscaching: simple and efficient negative sampling for knowledge graph embedding," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 614–625, IEEE, 2019.

[35] Y. Zhang, Q. Yao, and L. Chen, "Simple and automated negative sampling for knowledge graph embedding," *The VLDB Journal*, vol. 30, no. 2, pp. 259–285, 2021.

[36] T. Pellissier Tanon and F. Suchanek, "Querying the edit history of wikidata," in *European Semantic Web Conference*, pp. 161–166, Springer, 2019.

[37] L. Schmelzeisen, C. Dima, and S. Staab, "Wikidated 1.0: An evolving knowledge graph dataset of wikidata's revision history," *arXiv preprint arXiv:2112.05003*, 2021.

[38] K. AlGhamdi, M. Shi, and E. Simperl, "Learning to recommend items to wikidata editors," in *International Semantic Web Conference*, pp. 163–181, Springer, 2021.

[39] T. Pellissier Tanon, C. Bourgaux, and F. Suchanek, "Learning how to correct a knowledge base from the edit history," in *The World Wide Web Conference*, pp. 1465–1475, 2019.

[40] T. Pellissier Tanon and F. Suchanek, "Neural knowledge base repairs," in *European Semantic Web Conference*, pp. 287–303, Springer, 2021.

[41] D. Fernández-Álvarez, J. Frey, J. E. Labra Gayo, D. Gayo-Avello, and S. Hellmann, "Approaches to measure class importance in knowledge graphs," *Plos one*, vol. 16, no. 6, p. e0252862, 2021.

[42] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[43] A. Thalhammer and A. Rettinger, "PageRank on Wikipedia: Towards General Importance Scores for Entities," in *The Semantic Web: ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 – June 2, 2016, Revised Selected Papers*, pp. 227–240, Cham: Springer International Publishing, Oct. 2016.

[44] P. Rosso, D. Yang, and P. Cudré-Mauroux, "Beyond triplets: hyper-relational knowledge graph embedding for link prediction," in *Proceedings of The Web Conference 2020*, pp. 1885–1896, 2020.

[45] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, M. Galkin, S. Sharifzadeh, A. Fischer, V. Tresp, and J. Lehmann, "Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[46] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[47] K. S. Jones, "Automatic indexing," *Journal of documentation*, 1974.

[48] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp, and J. Lehmann, "PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings," *Journal of Machine Learning Research*, vol. 22, no. 82, pp. 1–6, 2021.

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[50] G. Vandewiele, B. Steenwinckel, T. Agozzino, and F. Ongenae, "pyrdf2vec: A python implementation and extension of rdf2vec," 2022.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[52] D. Fernandez-Álvarez, J. E. Labra-Gayo, and D. Gayo-Avello, "Automatic extraction of shapes using shexer," *Knowledge-Based Systems*, vol. 238, p. 107975, 2022.